

Zhen Yang

DEEP LEARNING METHODS FOR PATIENT PHENOTYPING FROM ELECTRONIC HEALTH RECORDS

Information Technology and Communications Sciences
Master of Science thesis
April 2019

ABSTRACT

Zhen Yang: Deep Learning Methods for Patient Phenotyping from Electronic Health Records
Master of Science thesis
Tampere University
Master's Degree Programme in Information Technology
April 2019

In this MSc thesis we employed convolutional neural network based architectures in classifying free-form discharge summaries from electronic health records in the Medical Information Mart for Intensive Care III database. We intended to investigate how well deep learning models can perform in patient phenotyping tasks using unstructured data.

We based our work on the previous work done by Gehrmann, Sebastian, et al. in their paper "Comparing deep learning and concept extraction based methods for patient phenotyping from clinical narratives". We performed our tasks first by replicating their results using slightly different implementation details, then we extended the network architecture they used in their work, and finally we compared the results of our architecture and their architecture.

The main work of this thesis is the extra sentence level network that we added to the network architecture we replicated. In our network architecture, we fed not only the word level but also the sentence level inputs to the networks, thus making the networks able to learn features from combinations of nearby sentences.

Our experiments have shown our network architecture had a better performance over the original network architecture. It gave better results on all the F1 scores for all phenotypes, we also saw an overall improvement on ROCAUC scores. This indicates that the networks can benefit from our sentence level input to better understand the unstructured data from eHRs.

Keywords: deep learning, convolution neural network, patient phenotyping, sentence level input

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

PREFACE

I would like to give my greatest thanks to my supervisor, Prof. Frank Emmert-Streib, for searching for the topic and data set, providing insightful ideas for my work, giving feedback during the writing process, and most especially for his patient guidance throughout the process of my work.

Thanks to Dr. Facihul Azam for his advise and helps with the algorithms.

Thanks to Han Feng for his help with the figure and the writing of my thesis.

The work in this thesis was an extension to the work by Gehrmann, Sebastian, et al., so I would like to thank them for publishing their codes and data set so that I could follow their work.

Tampere, 29th April 2019

Zhen Yang

CONTENTS

1	Introduction	1
2	Theoretical background	4
2.1	Electronic health records	4
2.2	Patient phenotyping	4
2.3	Artificial neural network	6
2.4	Deep learning	8
2.5	Comparative study of machine learning methods on eHRs	10
3	Methodology	13
3.1	Convolutional neural network	13
3.1.1	Concept of convolution	13
3.1.2	Review of CNN architectures	14
3.1.3	Basic architecture of CNN	15
3.1.4	Forward and backward pass of CNN	21
3.2	Data	23
3.2.1	MIMIC-III database	24
3.2.2	Discharge summaries from MIMIC-III	25
3.2.3	Annotated dataset	26
3.3	Processing pipeline	28
3.3.1	Preprocessing for word level input	28
3.3.2	Tokenization, indexing, padding	30
3.3.3	Preprocessing for sentence level input	32
3.4	Word embedding	35
3.5	Network architecture	36
3.5.1	Basic structure	36
3.5.2	Details of the practical network	38
4	Results and Discussion	41
4.1	Statistical analysis on the data	41
4.2	Network for word level input	43
4.3	Network for word and sentence level inputs	46
4.4	Comparing the two networks	47
4.5	Experiments on removing less frequent tokens from dataset	49
5	Conclusion	52
	References	55

LIST OF SYMBOLS AND ABBREVIATIONS

ANN	artificial neural network
CNN	convolutional neural network
CPU	central processing unit
eHR	electronic health records
GPU	graphics processing unit
NLP	natural language processing
exp	exponential
\mathbb{R}	set of real numbers
\in	is member of
\int	integral expression
∂	partial derivative
\pm	followed by standard errors
Σ	sum expression

1 INTRODUCTION

Electronic health records (eHRs) refer to the information that is systematically collected by clinical recorders from patients regarding their health information in digital form. eHRs contain varying categories of data, ranging from structured data such as diagnoses, laboratory results, and medications, to unstructured data like clinical texts. eHRs have been evolving significantly since 1980s and now have become a center stage in most European countries' national health informatics strategies [29]. The percentage of adoption of eHR systems rose from 9.4% to 75.5 % in US from the years 2008 to years 2014 [6]. Currently, eHRs are being rapidly applied, and many hospitals hold a basic eHR system.

eHRs are able to minimize the downsides of traditional paper records. They can be easily transferred between organizations through the internet. They can also effortlessly retrieve the patients' previous medical records and health information. Normally, eHRs are recorded with the information that is validated by the professionals. eHRs can be utilized in planning patient care, improving decision-making for patient care, management, and health policy. The departmental information such as intensive care records, ambulatory records, and emergency department records in eHRs already have been used for a period of time in improving the outcomes of the care programs [22]. Overall, in recent years, there is a considerable increase of usage of eHRs in researches, primary, secondary, and tertiary care domains.

With the rapidly increasing adoption rate of eHRs, data sets that contain rich eHRs information are available for researches. There are many different databases of eHRs that can be freely accessed such as the Medical Information Mart for Intensive Care (MIMIC) database, and the Informatics for Integrating Biology and the Bedside (i2b2) datamarts. These databases can be used for researches and applications, especially for the propose of secondary study. Recent studies have proven that secondary study and the use of eHRs can advance clinical research as well as better inform clinical decision-making [11, 43, 53]. Moreover, as computation biology is getting more and more attention by the data scientists these days [15], the utilization of eHRs in computation biology can help accelerate the process of applying computation biology.

Patient phenotyping is a classification task which aims to predict whether a patient has a specific medical condition or is suffering from a high risk of developing one [17]. eHRs contain numerous information regarding patient phenotype in both structured and unstructured data. Studies have shown that patient phenotypes inside electronic health records can help perform genome-wide association study [34], large-scale health re-

search initiatives [25], as well as identify adverse drug events [38].

Therefore, correctly deriving patient phenotypes from eHRs is essential for performing phenotype related tasks. Inside eHRs, the information is organized in parts structured form, and these data do not require sophisticated machine learning or statistical methods for their processing. As for the information in free text form such as clinical documents, these data contain the most abundant and essential information [44]. Free-form writing allows non-structured description which is often valued by the clinicians, and is hard to replace with structured forms. However, it is difficult to process the unstructured data due to their high heterogeneity, lack of standard grammar rules. Meanwhile the texts in unstructured data are full of acronyms, abbreviations, spelling and typing errors, not to mention that styles of the writing can be very different according to author-specific idiosyncrasies. This imposes the difficulty for free-form documents to be analyzed by computers. It is time-consuming to manually extract information from the unstructured data [25]. Thus natural language processing plays an important role to help us analysis free-text data.

Natural language processing (NLP) is a sub-field of artificial intelligence. Its goal is to enable computers to understand and process natural languages as close to human-level as possible. Usually, the workflow of NLP involves detecting the boundaries between each word, normalizing each word into its original form, tokenizing each word within boundaries into different tokens, part-of-speech tagging each token, and parsing the sentences. Making the computers interpret human languages is hard. There are many difficulties like segmenting words, sentences or even paragraphs, and understanding words and sentences that are ambiguous in different speaking contexts. However, with the help of machine learning techniques and NLP, many researches have shown promising results in medical NLP tasks, such as machine learning approach based personalized medicine study [12], and text mining in genomics [3].

Deep learning is a newly active subject in machine learning. Deep learning proposes many machine learning architectures that are able to outperform traditional machine learning algorithms. By applying deep learning algorithms we were able to see some dramatic improvements in the traditional NLP tasks, such as convolutional neural network on text categorization [28], convolutional neural network for sentence classification [30], and recurrent neural network for sentiment classification [52]. Compared with the traditional machine learning algorithms, deep learning architectures use artificial neural network, and usually refer to the machine learning algorithms that utilize more than one layer in their network architectures. By feeding the input sequences through multiple layers, the models are able to transform the input into a more informative representation of original data. In principle, deep learning provides models that are able to better exploit high-dimensional data sets by training networks with deep structures that can capture the internal patterns and higher-level features from the data [1]. Deep learning architectures include deep neural network, convolutional neural network, recurrent neural network, autoencoders, deep belief network, etc. They have been applied into several artificial in-

telligence sub-fields such as computer vision, natural language processing, and audio processing. These applications achieved competitive results over traditional methods.

For patient phenotyping from eHRs, traditional methods such as Mayo Clinical Text Analysis and Knowledge Extraction System (CTAKES) [48], MedEx [56], MetaMap [2], and Medlee [16] are developed to extract medical related terms from free-text clinical notes. They work in such a way that they identify phrases corresponding to certain medical entities in the texts [10], and use them as inputs to a predictive model. They rely heavily on a number of expert-defined medical concepts, also improving these algorithms normally requires time investment of experts in defining entities. However, combining deep learning and NLP enables the model itself to learn abundant representations of data. These representations can be later leveraged to learn which phrases in the texts are the most relative to some given phenotypes. Therefore, deep learning methods normally do not require hand-craft inputs, thus lessening the requirement of domain experts' intervention in the tasks, and deep learning models can be easily transferred [17]. In contrast deep learning methods also have disadvantage like their results are very hard to be interpreted. This imposes a big issue since being able to perform analyses on the data is the most important for data scientists to gain reliable knowledge and derive insights from the data sets and the methods [13].

Convolutional neural network (CNN) is a variant of deep learning algorithms. It utilizes convolution layer and pooling layer in its architecture. CNN has remarkable performance over other architectures in computer vision field since convolution layer is extremely good at extracting features from images. Recently CNNs have been successfully applied in NLP field. The researches included modelling and summarising documents [9], classifying sentences [30], and text categorization [27]. CNN also shows excellent performances in medical NLP tasks such as patient phenotyping [17], risk prediction [7], and disease prediction [50]. CNN has been suggested to be good at extracting local information regardless of their positions in the text [57].

In this thesis, we focused on deep learning methods especially on using a CNN architecture to perform patient phenotyping from clinical narratives. We followed the previous work by Gehrmann, Sebastian, et al. [17]. In their work, they used CNN to perform patient phenotyping on 10 different phenotypes from clinical documents in eHRs. The network architecture they applied had a downside of only searching adjacent words up to 5 neighbors, therefore, in our experiment, we added another input features called sentence level input by pooling all the word embeddings in each sentence, and this results in a single vector that we called sentence embedding. Additional, we modified the network architecture by adding another network that processes at the sentence level inputs, thus making the network architecture capable of considering the relations of adjacent sentences. Our network architecture has shown some improvements over the original network. For the results and conclusions, we compared and reported the error measures produced by both original and our network architectures regarding evaluating binary tasks [14].

2 THEORETICAL BACKGROUND

In this chapter, we are going to discuss the background related to the work in this thesis. We firstly talk about what are eHRs, and what is patient phenotyping. Then we will introduce artificial neural network (ANN) which is the fundamental structure of CNN. After that we will go through the concept of deep learning and its recent achievements. At last, we will talk about some of the natural language processing applications on eHRs.

2.1 Electronic health records

eHRs are digital forms of data including all aspects of patient health care information. eHRs comprise structured data such as medications, laboratories results, medical imaging data, and unstructured data like free text clinical notes. In order to store the data in a computer-readable form, eHRs contain data that are represented according to their relevant controlled vocabularies. These vocabularies contain standard identifications for different medical concepts such as Logical Observation identifiers Names and codes (LoINC) for laboratory result, and Digital Imaging and Communication in Medicine (DICOM) for imaging data. As for the unstructured data, there are no such overall standards but still some formal identifications like International Classification of Disease-9 (ICD-9) or ICD-10 are considered. eHRs were originally designed to help the hospital perform administrative tasks [25]. With the help of Health Information Technology for Economic and Clinical Health Act of 2009, the adoption rate of eHRs has skyrocketed for the past 10 years [5]. More and more studies focusing on secondary use of eHRs have been performed and showed good results [11, 43, 53]. eHRs can be used by hospital and clinics to improve patient care outcome and patient safety while providing rich resources for researches [32]. However, eHRs are difficult to mine due to their heterogeneous components and high-dimensional structures. Analyses on eHRs most of the time cooperate natural language processing techniques with machine learning algorithms. In this thesis, we focused on the unstructured data in eHRs, i.e. the clinical notes.

2.2 Patient phenotyping

Patient phenotypes are the different predefined criteria that a patient meets. The criteria are defined by medical concepts which can be the symptoms a patient has. The task

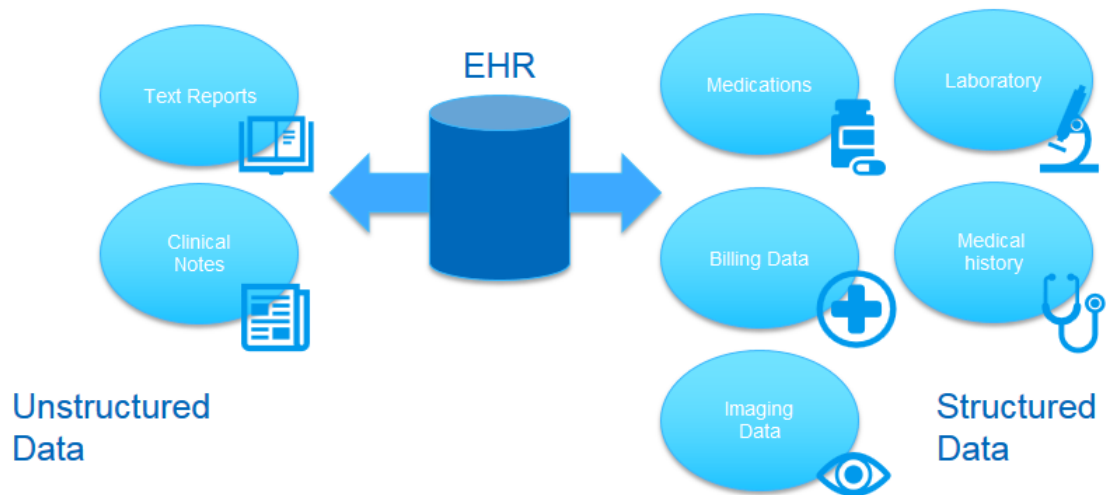


Figure 2.1. Basic components in eHRs.

of patient phenotyping is to correctly predict whether a patient has a specific medical condition or is under the risks of developing one. Therefore, properly deriving patient phenotypes from existing database is essential for performing patient phenotype related tasks which includes improving patients care, and carrying out many medical related researches.

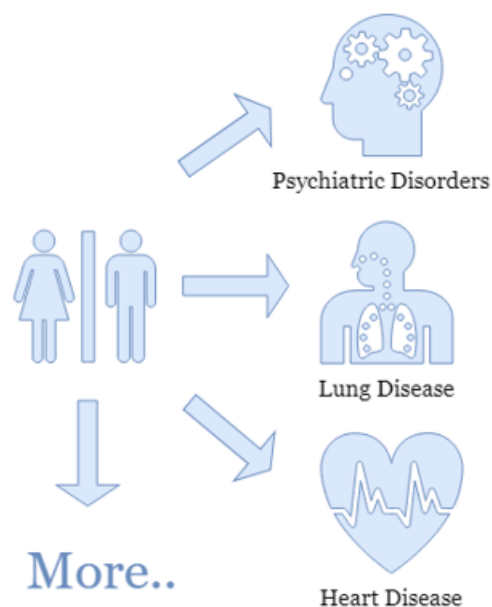
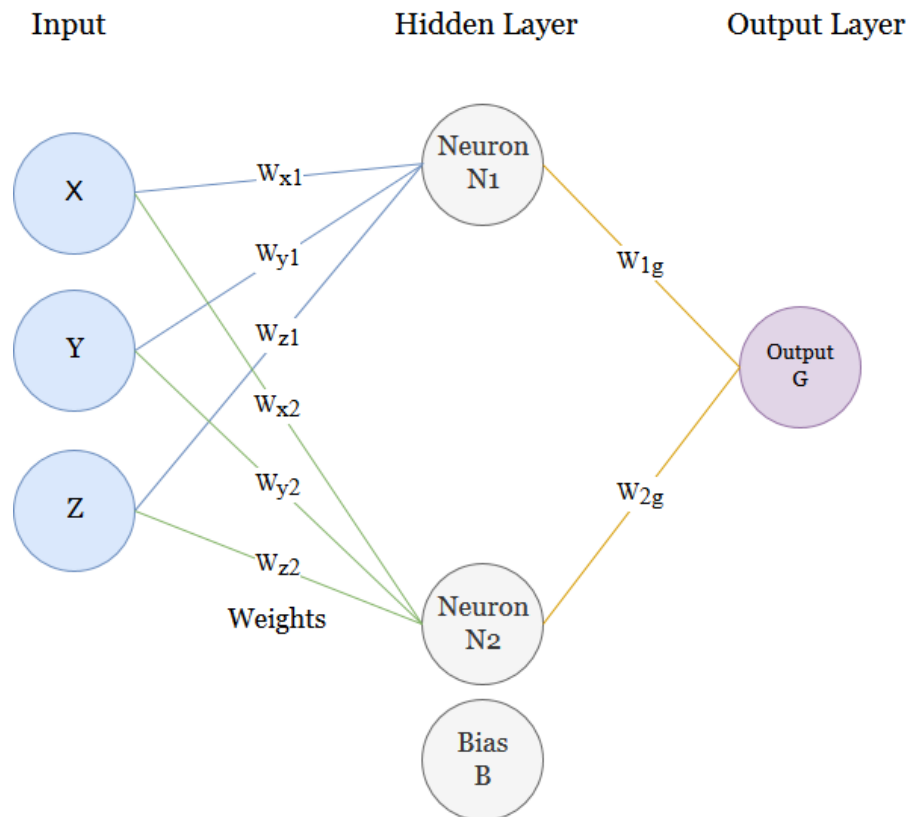


Figure 2.2. Possible phenotypes for patients.

2.3 Artificial neural network

Artificial neural network (ANN) is a method from in machine learning. ANN resembles the biological neural networks inside animal bodies [19]. In order to mimic the behaviors of biological neural networks, ANN contains circuits of neurons to be activated by the inputs. A fundamental structure of ANN comprises input layer, hidden layer and output layer. Hidden layers normally can be made up of multiple layers than one. An illustration of ANN structure can be seen in figure 2.3.



$$N1 = f(X * W_{x1} + B) + f(Y * W_{y1} + B) + f(Z * W_{z1} + B)$$

$$N2 = f(X * W_{x2} + B) + f(Y * W_{y2} + B) + f(Z * W_{z2} + B)$$

$$G = f(N1 * W_{1g}) + f(N2 * W_{2g})$$

$$f(\text{activation function}) = \text{linear } (f(x)=x)$$

Figure 2.3. The basic structure of an artificial neural network which has three basic layers, i.e. input, middle (hidden), and output layer. Each connection line represents an individual weight.

Like other types of machine learning algorithms, the goal of neural network is to learn hidden patterns inside the data according to the rules established by the network itself. However, unlike other machine learning algorithms which either rely much on the human-defined features, or are unable to fit complex data set, the power of ANN is that with enough neurons in the network it can be capable of learning any sophisticated patterns from the data with little human intervention. ANN is designed to calculate the loss be-

tween the results and the desired outputs, and update the weights in the network to optimal values using the loss. The network calculates the output by passing the inputs through one or more layers and eventually to output layer. During this process, the inputs are modified by the weights and the activation functions. The intermediate results are stored in the neurons, and the neurons with stored values become the inputs for the next layer. After the network obtains the output from the last layer, the loss can be calculated by the predefined loss function using the calculated output and the desired output that we provided. This whole process is called a forward pass of a neural network.

The weights stored in the network enable the network to learn complex patterns from the data. Therefore, the objective of the network is to modify these weights until they are able to minimize the loss to a predefined level. The loss is passed back through the network after a forward pass using a process called backpropagation. During backpropagation, the network will update each weight using the gradient regarding each weight. Gradient descent algorithms are mostly used in backpropagation to calculate the gradient of the loss function for each weight. The weights are updated by a predefined optimizer using a step size and the gradient each weight receives.

Let us assume the weights in the network as $w = (w_1, w_2, w_3, \dots, w_x)$, then the goal of the network is to model a relation $o = f(x, w)$, where x is the input, w are weights, and o is the real output. The network tries to make the calculated output which is modeled by $f(x; w)$ as close to the real output o as possible.

If the real output o and the input x were given, the loss term L of the network can be formulated according to the predefined loss function l . In machine learning, the most commonly used two loss functions are mean squared error loss and cross-entropy Loss. An example formula of calculating the loss term L can be calculated in equation 2.1.

$$L(w) = \frac{1}{n} \sum_{i=1}^n l(o_i, f(x_i; w)) \quad (2.1)$$

In order to find the minimal loss term L , one can use gradient descent to find the optimal values for weights. Chain rule is normally applied in calculating the gradient of the loss function to make the calculation efficient. Taking the figure 2.3 for example, if one wants to obtain the gradients for W_{x1} , the chain rule can be illustrated in the equation 2.2.

$$\frac{\partial L}{\partial W_{x1}} = \frac{\partial L}{\partial G} \frac{\partial G}{\partial N_1} \frac{\partial N_1}{\partial W_{x1}} \quad (2.2)$$

The weights are updated using formula defined by optimizer. The most commonly used optimizers are Adam [31] and Adadelta [58]. Let's assume for each weight w^n , the learning rate is γ , then the common formula for updating each weight can be defined in 2.3.

$$w_{\text{new}}^n = w^n - \gamma \frac{\partial L}{\partial w} \quad (2.3)$$

Activation function is one of the critical roles in neural network. It helps improve the expressive power of the network. Activation function is normally applied before passing a value into a neuron. The formula of applying activation functions is usually denoted as $y = f(W^T x + b)$, here b is the bias term, f is the activation function, y can be a neuron and x is the input. Activation function transfers the value into a new one according to different activation types. Table 2.1 shows some common activation functions.

Table 2.1. Some common activation functions.

Name	Equation	Range
Logistic (Sigmoid)	$f(x) = \sigma(x) = \frac{1}{1 + \exp^{-x}}$	(0,1)
TanH	$f(x) = \tanh x = \frac{\exp^x - \exp^{-x}}{\exp^x + \exp^{-x}}$	(-1,1)
Rectified linear unit (ReLU)	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$[-1, \infty)$

2.4 Deep learning

Deep learning is a subset of machine learning, and machine learning is part of artificial intelligence. The goal of machine learning is to design statistical models that can derive underlying patterns from the data while keeping the models able to alter themselves when they are exposed to new data. Hence, the models can adapt to new data without intervention of human.

The conventional machine learning models need to transform the raw data into an internal representation that can be recognized by the classifier. Hence the patterns could be detected by the models from the input data. In order to process data in their raw form, traditional machine learning methods implicitly rely on careful engineering and a considerable amount of intervention of domain expertise [35]. Therefore, these models are limited to their shallow structures.

Deep learning algorithms are essentially any neural networks that consist of more than one hidden layers. While each hidden layer is supposed to transfer the input into a more abstract representation of the raw data, the network can learn very complex functions with the combination of enough non-linear modules. The most important fact is that all these learning procedures are performed by the network itself independently. Thus deep learning can also be considered as representation learning techniques. Deep learning is able to improve over the traditional machine learning techniques where human efforts are needed in constructing new rules for learning patterns. Therefore, it has been more and more favoured by the researchers recently.

There are mainly three types of deep learning methods: supervised learning, unsupervised learning, reinforcement learning. Their definitions can be seen in the table 2.2.

Table 2.2. Three main types of deep learning categories.

Methods	Definition
Supervised Learning	The labels are known for all the existing samples for training. The objective of the network is to modify the parameters in the network according to the loss between the predicted label and the true label. After sufficient training process, the network is able to correctly generate labels for unseen data.
Unsupervised Learning	All the data for training the network is unlabeled. the objective of the network is to divide the data into different clusters where there are one or more common characteristics. The network doesn't produce any correct outputs, instead it explores the data and derives inner structures and relations from the data.
Reinforcement Learning	In reinforcement learning, there is no labeled data at the beginning of the training, but the labeled data are generated during training. The algorithm tries to make the best decision for each action, and each action is rewarded or punished according to the generated rules during training.

Many deep learning architectures have been proposed these days, and they all have superior performances in specific domains. In the following texts we will go through a few most representative algorithms.

Multilayer perceptrons is a classical type of ANN that consist of multiple hidden layers. All the layers in multilayer perceptrons are fully-connected, and in this case every neuron is connected to every other neuron in the next layer. Each node inside the network uses a neuron with a non-linear activation function. The network is expected to capture more abstract information from the input through multiple stacked layers, but this type of network is often limited due to its simple structure, one direction only flow of data, and huge amount of parameters requirement.

Recurrent neural network which utilizes directed graph inside the network is able to cap-

ture the temporal behaviors from a sequence input. It has shown excellent performance on sequential data, like audio, and text [21, 54].

Autoencoder tries to learn the essential representations of data using an encoder network and a decoder network in an unsupervised manner. Encoder reduces the dimensions of input, while decoder attempts to reconstruct the data compressed by the encoder back to its original input as similar as possible. During this process the network updates the parameters to improve itself until it is able to reproduce the representations that capture the critical parts from the inputs. Decoder and encoder can use different types of network architecture.

Deep belief network denotes networks that stack unsupervised network architectures as its basic components. Usually its components can be restricted boltzmann machines or autoencoders. By initializing the model using the unsupervised networks deep belief network learn its parameters one inner network by one. Deep belief network can be used to perform supervised task by stacking an output layer to the last layer. Due to the fact that the network already learned the parameters during the unsupervised phase, the network can further fine-tune the parameters during the supervised task thus producing promising results.

Convolutional neural network is a type of network using one or more convolution layers and pooling layers to extract different representative features from the inputs. The network makes prediction based on the learned features. CNN has been dominating computer vision field, and many of its variants have been proposed to further push this trend [35]. In recent years CNN has also shown excellent performance in natural language processing field [7, 30, 50]. It has been suggested to be good at extracting local position-invariant features from the input for classification tasks [57]. Therefore, this thesis focused on how to utilize CNN to perform NLP classification task.

2.5 Comparative study of machine learning methods on eHRs

eHRs hold rich resources for medical researches, a number of studies have been performed on eHRs. In order to explore eHRs efficiently, utilizing machine learning algorithms is essential. Past studies mostly used traditional machine learning algorithms. Although these days more and more studies have been carried out using deep learning algorithms. Deep learning algorithms were proven to have better results compared with the traditional methods for some tasks [17]. They also require lesser intervention of human experts because they don't rely on heavy hand-craft features from expertise. They are able to learn high-level abstract representations from the data by themselves. With the growing amounts of available training data, deep learning algorithms will be increasingly demanded in processing eHRs. Not to mention that there are new deep learning algorithms that achieve state-of-the-art performances coming out at times, which will also help accelerate the process of applying deep learning algorithms [35]. Since we are

dealing with unstructured data in this thesis, we will only review and discuss some of the researches performed on the unstructured data from eHRs. The methods to be discussed ranging from traditional machine learning to modern deep learning methods.

Mayo clinical text analysis and knowledge extraction system (cTAKES) [48] is an open-source natural language processing system that focuses on extracting information from unstructured digital medical records. It was proposed by Savova, Guergana K., et al in 2010 [48]. It aims to process clinical narratives from electronic health records by recognizing and annotating medical related terms in the texts. cTAKES consists of a system of pipeline components. These components include sentence boundary detector, tokenizer, normalizer, part-of-speech tagger, shallow parser, named entity recognition annotator, status annotator and negation annotator. The system processes input by applying the components mentioned above in sequence, and outputs a structure that contains information about all the recognized and annotated entities along with some attributes marking their properties. These structured features can later be used as input to predictive models. cTAKES relies on rule based and machine learning techniques to extract information from clinical notes. Each of its components achieves comparative results. All together they produce a promising solution to extracting information from unstructured clinical notes.

In Zhou, Li, et al their work [61], they performed a study on comparing different methods for identifying patients with depression from discharge summaries. In their paper, they used NLP technique combined with traditional machine learning algorithms, and they used 1,200 randomly selected patients with discharge summaries. The data set was annotated into three categories: high confidence, intermediate confidence, low confidence. They processed the data by firstly applying a NLP system called MTERMS [60] to extract the related terms for depression symptom, and these terms were later used as features to the classification algorithms. They compared the performances between MTERMS decision tree, SVM, NNge, RIPPER, and C4.5 decision tree. MTERMS decision tree was reported to have the best F1 scores over all other algorithms. Their work has shown that traditional machine learning methods can perform well on the classification task based on unstructured data. However, traditional methods typically rely heavily on hand-craft features defined by experts. Algorithms they employed were unable to understand the terms that were outside the scope of predefined medical terms. Thus these algorithms can not utilize those undefined terms which might be essential for predicting.

In Geraci, Joseph, et al their work [18], they used deep learning methods to handle unstructured data from eHRs. Their objective was to predict whether a patient is qualified for recruiting for depression study. In their work, they annotated 861 patients according to their clinical notes extracted from eHRs. They built two multilayer feed-forward deep neural network architectures. The first had specificity 97% and sensitivity 44.5% while the second one had specificity 53% and sensitivity 89%. They combined two networks by passing the results from the first network to the second network, thus producing a result of specificity 87% and sensitivity 75%. Their work showed that neural networks

with even simple feed-forward architecture is able to perform well in classification task on unstructured data. Additionally, their research has shown the network architectures that scientifically combines two different neural networks together can improve the overall performance.

In Gehrmann, Sebastian, et al [17], they proposed a CNN based method on patient phenotyping task from discharge summaries. In their work, they replicated the CNN architecture from Kim Yoon [30]. They trained their network on 1,610 patient discharge summaries extracted from MIMIC-III [26] database, and all 1,610 samples were labeled into 10 different phenotypes. They compared the performance of CNN with some baseline models, and it turned out that the CNN model constantly outperformed other baseline models. Moreover, they interpreted their model by extracting the most predictive phrases from CNN. It turned out that CNN is able to detect some difficult task-related phrases that are even hard to be interpreted by non-experts. Their work has shown that a suitable deep learning algorithm is able to outperform the traditional machine learning algorithms by large margins. While traditional machine learning methods count on human predefined medical related terms, deep learning methods can help expertise to save their efforts on defining hand-craft features. This paper motivates the work of this thesis, also their paper is the main reference and previous work that this thesis based on. Our work replicated the network architecture, and we slightly modified the network they used by feed additional features to an another CNN.

In summary, we have seen a trend in NLP from using statistic rule-based systems to traditional machine learning methods to using deep learning methods. Most of the research now has been focusing on using machine learning especially deep learning methods to analyze medical records. We believe that with the growing of available eHRs and the vast spawning rate of novel machine learning algorithms, deep learning will be more and more commonly applied in dealing with medical records.

3 METHODOLOGY

In this chapter, we will discuss all of the details of our network architecture, the data set we used, and all the preprocessing steps regarding how we obtained our input data including word level and sentence level inputs.

3.1 Convolutional neural network

In this section, we will go through the basic ideas and components of CNN. We will discuss the fundamental components of CNN individually, and then we will consider the training process of CNN.

3.1.1 Concept of convolution

This subsection introduces the concept of convolution, and why we use convolution in the neural network.

In mathematics, a convolution operation can be interpreted as the amount of overlapping area of one function g shifting over another function f [55]. Alternatively, it can be explained as that an output of a system at a time can be formulated as the total impacts of current and previous inputs. The formula of a typical convolution between functions f and g over a finite range $[0, t]$ is given by:

$$[f * g](t) = \int_0^t f(\tau)g(t - \tau)d\tau \quad (3.1)$$

Where $[f * g]$ is denoted as convolution between functions f and g [55], here g is the convolution kernel that applied to the function f .

In image processing field, a convolution operation can be considered as a dot production between the matrix of a group of pixels and the matrix of convolution kernel. After applying one convolutional kernel to all the pixels in the image, the result will be a matrix with certain width and height depending on the spatial arguments (we will talk about them in later sections). This matrix can be considered as a group of weighted means of the corresponding pixels. Different kernels can be used to enhance an image in different ways. For instance, the Laplacian kernel which aims to sharpen the image can be defined as in

figure 3.1.

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 3.1. Two common Laplacian kernels

By applying convolution operation between this kernel and an image we can sharpen the image to get more details about the edges in the image. Similarly, one can change the weights of the kernel to extract varying features from the image.

In practice, if we apply the concepts of convolution in the neural network, then different kernels can detect different figures in an image. The network can learn as many features as the number of the filters (kernels) by using convolutional kernels in the network, and eventually make decisions according to the features learned by the network. This type of network turned out to be very successful in the image classification and recognition task in machine learning [33].

3.1.2 Review of CNN architectures

CNN is an analogous architecture of the general ANNs. The way that CNN differs from other architectures is the convolution layer and the pooling layer. These layers acts as the heart of CNN. CNN utilizes the convolution operation to extract certain patterns from the input.

LeNet-5 [36], which was invented by Yann LeCun in 1998, is known to be the first model that introduced convolution and pooling layers into the network. Although because of the lack of training data, the insufficient computer's processor speed, the model did not perform well at that time. Nevertheless, their paper established the basic components of CNNs . It was not until 2012 did people come to realize how powerful CNN can be in image related task. In the ImageNet 2012 competition, Alex Krizhevsky along with his AlexNet [33] won the first place in the image classification task with 15.3% error rate while the second place only reached an error rate of 25.2%. AlexNet had dramatically advantaged the traditional approaches, and the emergence of AlexNet has triggered the new era of learning in CNN. Ever since the achievement of AlexNet, more and more researches and efforts have been put into CNN. There are several remarkable revolutions of CNN, such as VGGNet [49], GoogleNet [51] and ResNet [23]. Further works involve the modification to the convolutional kernels, and improvement to the structure of the network. They all aim to make the network smaller and more flexible, while improving the performances.

3.1.3 Basic architecture of CNN

A typical CNN architecture includes convolutional layer, pooling layer, fully-connected layer 3 basic components [33, 36, 59] as in figure 3.2.

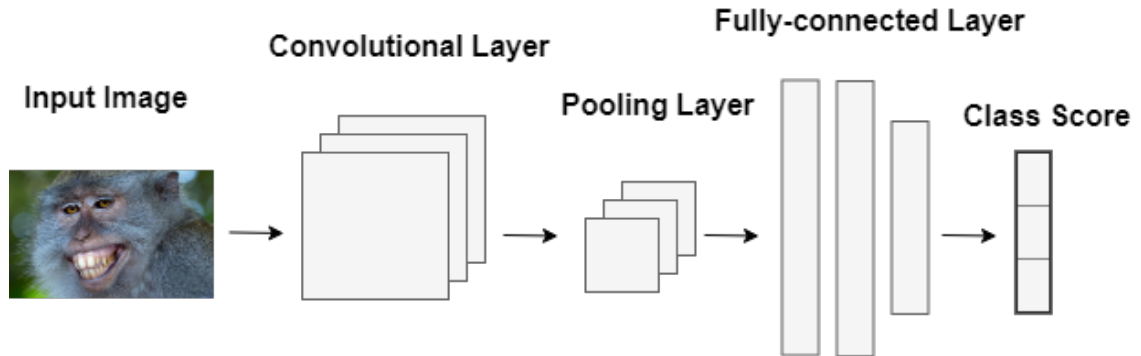


Figure 3.2. Basic architecture of CNN.

Overall, CNN takes an input and pass it through several feature extractors, and eventually transforms the features it learned to the probabilities of the classes.

Convolutional layer

For training large images, tradition neural network is limited due to the large amount of parameters in the network. Assuming we have an image with 500x500 pixels, in the hidden layer we have 100 neurons, then the total parameters of this layer will be $500 \times 500 \times 100 = 25M$, but this is only a single layer. As the network goes deeper, the numerous number of parameters will make the network impossible to train. Therefore, tradition neural network is nearly incapable of building a deep structure for image processing task. In contrast, the parameter requirement has been much lessened in CNN. The convolution kernel enables local connectivity and parameter sharing, which hugely reduce the parameters needed in the network. These properties make it possible for building larger and deeper network toward image machine learning tasks.

The convolutional layer is where the input will be processed with sliding kernels going through the whole specified dimension. This process will produce feature maps that contain certain features. Some spatial arguments are needed in this layer to generate fixed size of feature maps.

- 1. Depth can also be referred to the number of filters. It specifies how many different convolutional kernels for a specific filter length will be used in this layer. If it is 100, it means in total 100 feature maps will be produced after processing the input.
- 2. Stride defines how many steps the filter will move to next position. If it is 1, the filter will move one pixel at a time. It can be specified by the users to achieve different sizes of feature maps. The larger the stride is the smaller the a feature

map will be.

- 3. Zero-padding(P) is the padding that is used to stack to certain dimension of the input. Sometimes it is necessary to specify how many zeros we want to pad to the border of the input image in order to produce feature maps with the same horizontal and vertical dimension as the input.

These 3 hyper-parameters help control the size of the outputs of the convolutional layer. The shape of feature maps generated by the filters can be calculated by equation 3.2. Assuming the input shape is $W_{input} \times H_{input} \times D$, then the output volume of feature maps can be calculated as $W_{out} \times H_{out} \times N$ in the equation 3.2.

$$\begin{aligned} W_{out} &= \frac{(W_{input} - K + 2P)}{S + 1} \\ H_{out} &= \frac{(H_{input} - K + 2P)}{S + 1} \\ N &= D \end{aligned} \quad (3.2)$$

Where K is the window size of the filter, S is the stride step, P is the number of zero-padding, and N is the number of filters.

An illustration of how convolution layer produces a feature map can be seen in the figure 3.3.

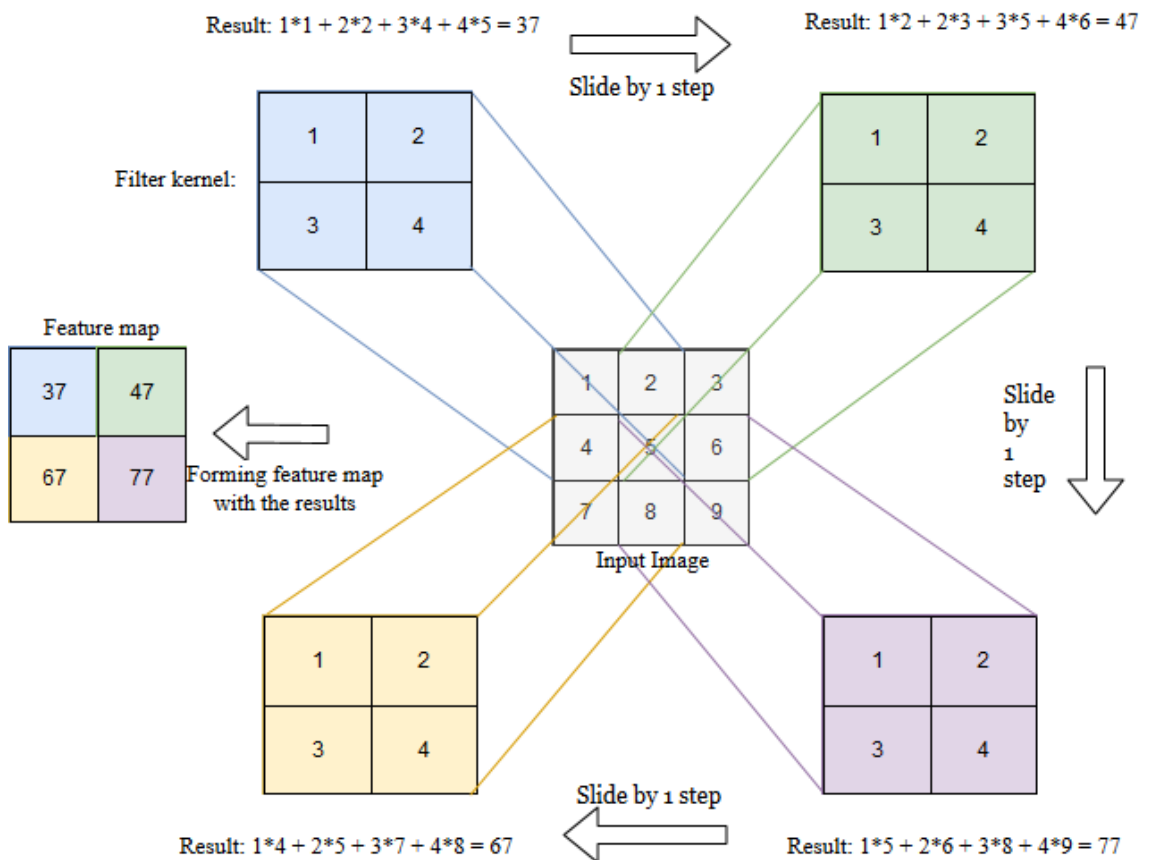


Figure 3.3. An example of how convolution layer operates on the input. In this case, one filter 2x2 with stride 1 processes the input and produces one feature map.

Sparse connectivity and shared weights

As it is mentioned above when we are dealing with large images, each pixel is connected to all the other neurons in the next layer. For a deep network the parameters will be so large that the model will be almost unable to train. However, if each neuron is only connected to a subregion of the input, the number of parameters will be significantly reduced. Furthermore, if we can share the weights for all the connections between a neuron and the local regions it connects to. We can even achieve better parameter requirement. These are the ideas of sparse connectivity and weights sharing in CNN. An illustration can be seen in the figure 3.4.

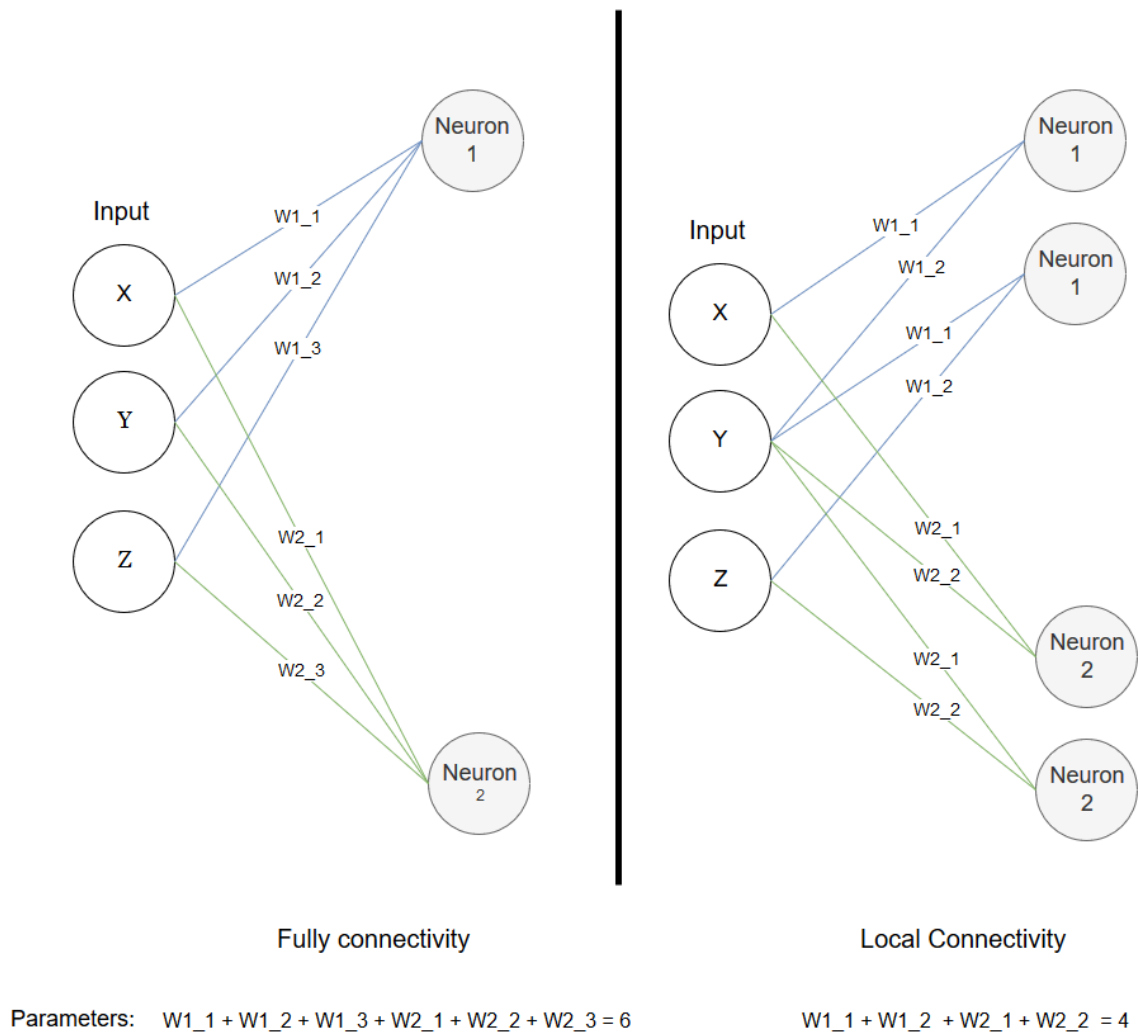


Figure 3.4. Illustration of local connectivity and weights sharing. On the left is the fully connectivity in normal neural network architecture. On the right is the local connectivity enabled by convolutional layer. In this layer the size of local region is 2, hence each neuron only connects to 2 input nodes at a time, and the weights are shared for a group of neurons

The convolutional kernel that produces the corresponding feature map will have a set of shared weights, and different kernels will have unique sets of weights. The region that a kernel is connected to is referred to as the receptive field. Each value in the feature map

has its receptive field from the original image. When dealing with two-dimensional images (width and height) with R,G,B channels, the connections of a filter to the image are local in the space of width and height, but to the depth of the total channels of the image. Therefore, each generated pixel in the feature map is resulted from the convolution of its receptive field across all the channels from the image. Assuming the input image has dimension $W \times H \times C$, the filter size is $S \times S$, and number of feature map is D , then each filter window has weights of dimension $W_{1:D} \in \mathbb{R}^{S \times S \times C}$.

Convolution operation

The convolution layer operates in the way that each kernel will slide through the whole image with the specified spatial arguments and a fixed filter size. It will produce feature maps that contain the dot production between the kernel and pixels at the responding positions. All the feature maps will stack along the depth dimension to form the final output of the layer. An example of convolution between an input image with 3 channels and a kernel is illustrated in figure 3.5.

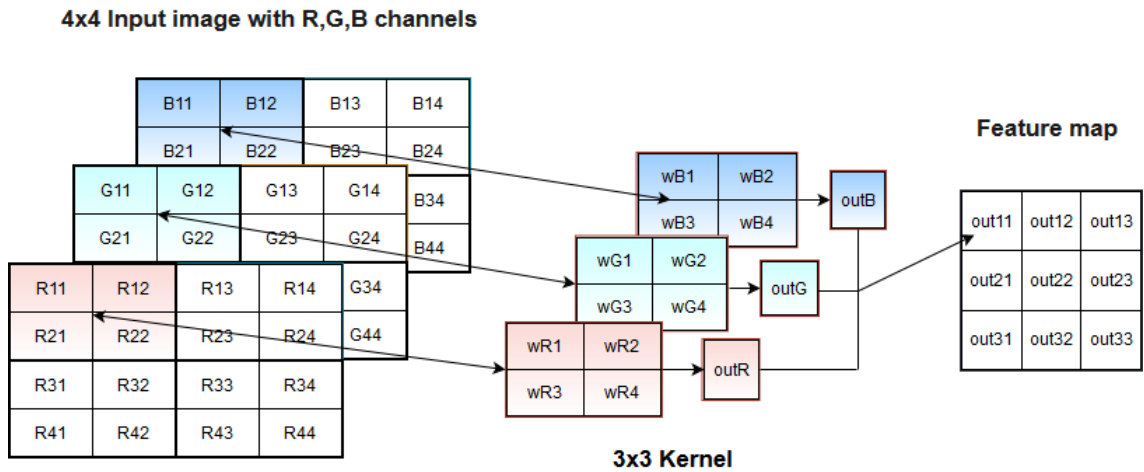


Figure 3.5. Example of convolution between an input image with R,G,B 3 channels and one 2x2 filter with stride 1 and 0 zero-paddings.

There is only one filter in the figure 3.5, so only one feature map will be produced. Value out_{11} in the feature map can be calculated according to the equation 3.3, and other values can be computed likewise. The bias neuron is 0 in this example, also in practice there should be an activation function applied to the results before assigning the final values to the feature map. For the simplicity here the activation is just identity mapping where $f(x) = x$, but possible activation functions to be used can be seen in table 2.1.

There are 3 channels for the input image in figure 3.5. The filter will have the third dimension same as the one of the input. Therefore, this filter will have three separated windows

and it has in total $2 \times 2 \times 3 = 12$ parameters in total.

$$\begin{aligned}
 out_{11} &= out_R + out_G + out_B \\
 out_R &= R_{11} \cdot wR_1 + R_{12} \cdot wR_2 + R_{21} \cdot wR_3 + R_{22} \cdot wR_4 \\
 out_G &= G_{11} \cdot wG_1 + G_{12} \cdot wG_2 + G_{21} \cdot wG_3 + G_{22} \cdot wG_4 \\
 out_B &= B_{11} \cdot wB_1 + B_{12} \cdot wB_2 + B_{21} \cdot wB_3 + B_{22} \cdot wB_4
 \end{aligned} \tag{3.3}$$

Pooling layer

Pooling layer is normally inserted after a convolutional layer, and it is used to downsample the data and pass the downsampled data to next layer. Pooling will result in outputs with reduced spatial sizes. The pooling layer requires some spatial arguments, i.e. the pooling window, stride and the zero-padding. For a 2d image with 3 channels, pooling will operate in each channel independently, therefore, does not effect the length of depth dimension. With a pooling windows of size 2×2 and stride 2 and 0 padding, we are equivalently downsampling the input by half by the height and width.

Pooling Window 2×2 , stride 2

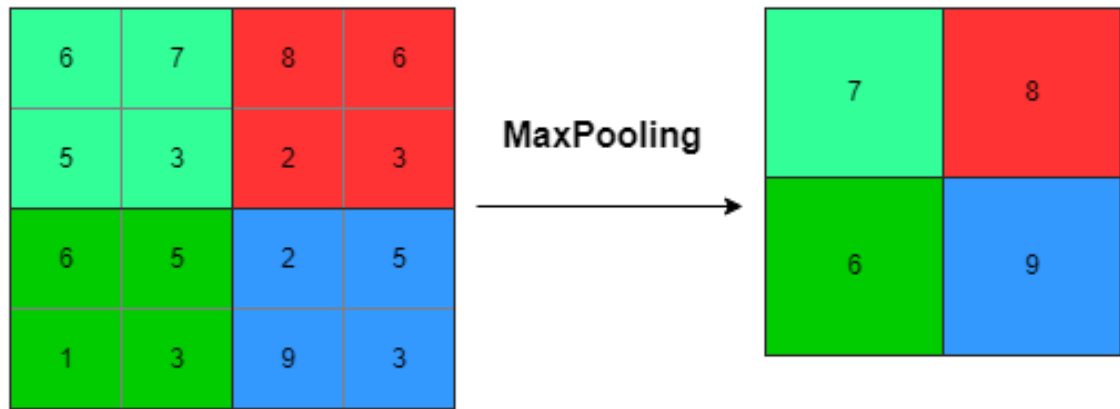


Figure 3.6. A pooling operation with 2×2 window and stride 2 on an input. Different colors represent the values pooled by the corresponding areas.

There are several pooling options, such as max-pooling, average-pooling, and sum-pooling. The values inside the pooling window will be calculated by the specified pooling method, which results in new values replacing the corresponding pixels, thus achieving downsampling of the input. In general, max-pooling is more commonly used in CNN architecture. There are some advanced pooling methods like stochastic pooling [59] and fractional max-pooling [20]. They have been showed to have better results than basic pooling methods. However, there is no single best method for all tasks, the method one should use is tasks depending.

Downsampling the input makes the network smaller and more flexible and easier to scale

with large input data. In addition to the downsampling, pooling can help achieve invariances including translation invariance, rotation invariance, and scale invariance [24]. It does not matter where the object is in the image, or how large the object appears in the image. We can get close results by performing max-pooling operation. This technique makes the model more robust to noises. In max-pooling, it only uses the weights which are the most informative, since small weights will not contribute much to final prediction of the network. This helps lower the dimensions of the inputs without losing performance. An illustration of invariances introduced by the pooling layer can be seen in figure 3.7.

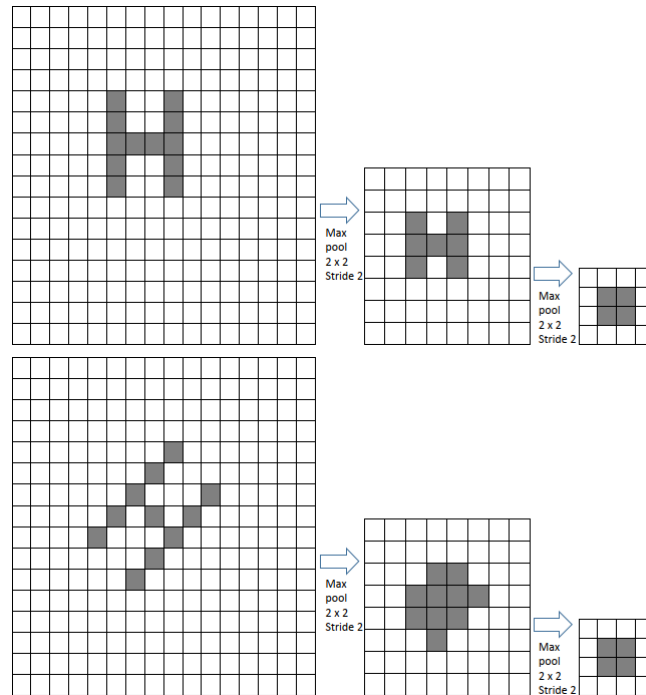


Figure 3.7. An example of invariance introduced by the pooling layer.

Fully-connected layer (FC)

In the traditional CNN architecture, we apply a FC layer in between penultimate layer and output layer. While convolutional layer and pooling layer map the input image into a collection of high-level features of data, FC layer will learn the non-linear combinations of these features by taking the weighted mean from the features.

However, FC layer uses fully-connectivity, which means that each of its neuron will connect to all of the neurons in previous and next layer. It is reported that the number of parameters introduced by FC layer can reach millions and it can take up to 80% of the total parameters in the network. With this huge number of parameters it can easily cause over-fitting [49]. Therefore, one trend of new CNN architecture is to build without the FC layer. Some approaches have been proposed to replace the FC layer in CNN. In the paper [37], they proposed a global average pooling method to replace the functionality of FC layer in CNN. This method reduces the amount of the parameters used in model

while achieving rather good results.

3.1.4 Forward and backward pass of CNN

As a variant of neural network, CNN also takes advantages from gradient descent and the backward-propagation to update the weights in the network. Since there are convolution and pooling layers in CNN, the training details may differ from traditional network, but it still follows the procedure by firstly calculating the loss, and passing the loss back to the network, then updating its parameters using the loss.

Forward pass of CNN

The forward pass for convolution layer includes calculating the convolution between the kernels and the inputs, which can be seen in the figure 3.8, and the outputs are calculated according to the equation 3.4.

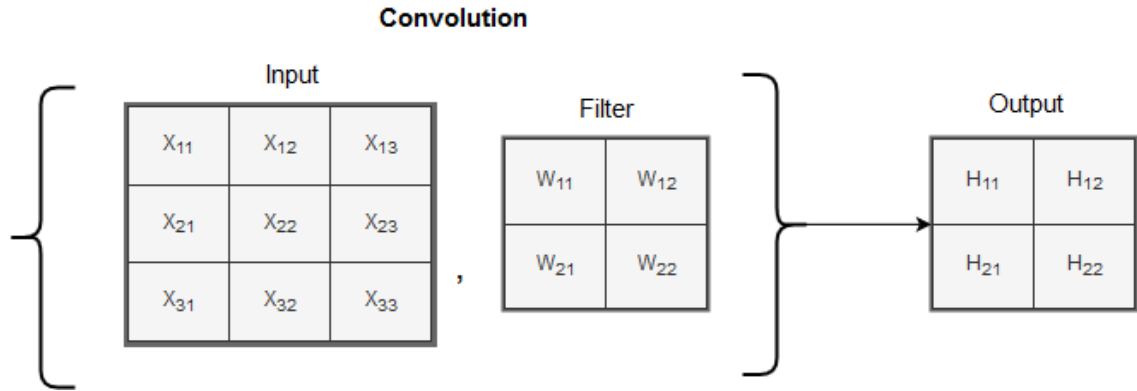


Figure 3.8. An example of forward-pass in a convolutional layer.

$$\begin{aligned}
 H_{11} &= X_{11}W_{11} + X_{12}W_{12} + X_{21}W_{21} + X_{22}W_{22} \\
 H_{12} &= X_{12}W_{11} + X_{13}W_{12} + X_{22}W_{21} + X_{23}W_{22} \\
 H_{21} &= X_{21}W_{11} + X_{22}W_{12} + X_{31}W_{21} + X_{32}W_{22} \\
 H_{22} &= X_{22}W_{11} + X_{23}W_{12} + X_{32}W_{21} + X_{33}W_{22}
 \end{aligned} \tag{3.4}$$

The forward pass of the pooling layer depends on the pooling type. For an illustration of max-pooling forward pass can be seen in figure 3.6.

Backward pass of CNN

The backward pass in CNN includes calculating the loss between the calculated output and the real output, passing the loss from the last layer back to every previous layer using

the chain rule, calculating the gradients for each weight, and then updating each weight according to gradient with respect to it using predefined optimizing algorithm.

Gradients pass to convolution layer

Assuming that the weights from the forward pass are calculated according to the equation 3.4 and we consider the example from figure 3.8. $H_{i,j}$ are the outputs from the convolutional layer, and they are also the inputs to the next layer. The gradients of $H_{i,j}$ can be computed as $\frac{\partial L}{\partial H_{i,j}}$, L is the loss term estimated from the loss function. According to the chain rule, the gradients for the weights in the filter can be computed as equation 3.5.

$$\begin{aligned}
 \frac{\partial L}{\partial W_{11}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial W_{11}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial W_{11}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial W_{11}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial W_{11}} \\
 \frac{\partial L}{\partial W_{12}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial W_{12}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial W_{12}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial W_{12}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial W_{12}} \\
 \frac{\partial L}{\partial W_{21}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial W_{21}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial W_{21}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial W_{21}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial W_{21}} \\
 \frac{\partial L}{\partial W_{22}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial W_{22}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial W_{22}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial W_{22}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial W_{22}}
 \end{aligned} \tag{3.5}$$

Similarly one can use the chain rule to calculate the gradients for input $X_{i,j}$ in equation 3.6.

$$\begin{aligned}
 \frac{\partial L}{\partial X_{11}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{11}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{11}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{11}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{11}} \\
 \frac{\partial L}{\partial X_{12}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{12}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{12}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{12}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{12}} \\
 \frac{\partial L}{\partial X_{13}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{13}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{13}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{13}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{13}} \\
 \frac{\partial L}{\partial X_{21}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{21}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{21}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{21}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{21}} \\
 \frac{\partial L}{\partial X_{22}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{22}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{22}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{22}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{22}} \\
 \frac{\partial L}{\partial X_{23}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{23}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{23}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{23}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{23}} \\
 \frac{\partial L}{\partial X_{31}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{31}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{31}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{31}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{31}} \\
 \frac{\partial L}{\partial X_{32}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{32}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{32}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{32}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{32}} \\
 \frac{\partial L}{\partial X_{33}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{33}} + \frac{\partial L}{\partial H_{12}} \frac{\partial H_{12}}{\partial X_{33}} + \frac{\partial L}{\partial H_{21}} \frac{\partial H_{21}}{\partial X_{33}} + \frac{\partial L}{\partial H_{22}} \frac{\partial H_{22}}{\partial X_{33}}
 \end{aligned} \tag{3.6}$$

Gradients pass from pooling layer

There are no parameters for the windows of pooling layer, hence during backward pass pooling layer does not need to update any weights. It only needs to pass the gradients from current layer back to previous layer. We can apply chain rule to the pooling layer, and the result will be different depending on different pooling method. We take max-pooling for example as shown in figure 3.9.

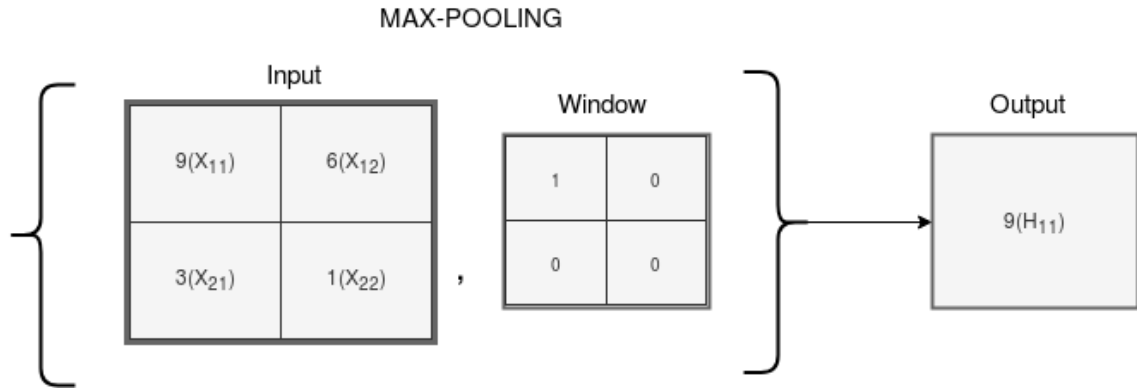


Figure 3.9. An example for forward pass of max pooling layer.

Output of pooling layer is calculated as:

$$H_{11} = X_{11} \cdot 1 + X_{12} \cdot 0 + X_{21} \cdot 0 + X_{22} \cdot 0$$

The gradients of the inputs can be computed as follow:

$$\begin{aligned} \frac{\partial L}{\partial X_{11}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{11}}, \quad \frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{12}} \\ \frac{\partial L}{\partial X_{21}} &= \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{21}}, \quad \frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial H_{11}} \frac{\partial H_{11}}{\partial X_{22}} \end{aligned}$$

Which then can be simplified to:

$$\begin{aligned} \frac{\partial L}{\partial X_{11}} &= \frac{\partial L}{\partial H_{11}} \cdot 1, \quad \frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial H_{11}} \cdot 0 \\ \frac{\partial L}{\partial X_{21}} &= \frac{\partial L}{\partial H_{11}} \cdot 0, \quad \frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial H_{11}} \cdot 0 \end{aligned}$$

Hence only the neuron that achieved the maximum value will get the gradients from next pooling layer.

3.2 Data

In this thesis, we used the dataset from [17] by Gehrmann, Sebastian, et al (2017). According to their work, the dataset was extracted from the discharge summaries in MIMIC-III database. There were 1,610 samples and these samples were annotated into 10

different phenotypes. We used the same data set in our experiments. In this section, we will introduce the MIMIC-III database. Then we will discuss and analysis the data set we used in this thesis.

3.2.1 MIMIC-III database

Medical Information Mart for Intensive Care (MIMIC-III) is a freely accessible database that contains eHRs information about ICU (Intensive Care Unit) for about 53,423 different hospital admissions for adult patients between year 2001 and 2012. This was collected from Beth Israel Deaconess Medical Center in Boston, Massachusetts. MIMIC-III is a powerful database since it is the only free and accessible critical care dataset, and it contains data collected for more than a decade. The information inside MIMIC-III is detailed and specific. MIMIC-III can be utilized in analysis and education around the world. Data in the MIMIC-III database ranges from structured data recorded using controlled vocabularies to free-text data such as clinical notes and text interpretations of images studies [26]. MIMIC-III consists of 8 different classes of de-identified data, which are shown in table 3.1.

Table 3.1. 8 different classes in MIMIC-III database [26].

Class of data	Description
Billing	Coded data recorded primarily for billing and administrative purposes. Includes Current Procedural Terminology (CPT) codes, Diagnosis-Related Group (DRG) codes, and International Classification of Diseases (ICD) codes.
Descriptive	Demographic detail, admission and discharge times, and dates of death.
Dictionary	Look-up tables for cross referencing concept identifiers (for example, International Classification of Diseases (ICD) codes) with associated labels.
Laboratory	Blood chemistry, hematology, urine analysis, and microbiology test results.
Medications	Administration records of intravenous medications and medication orders.
Notes	Free text notes such as provider progress notes and hospital discharge summaries.
Physiologic	Nurse-verified vital signs, approximately hourly (e.g., heart rate, blood pressure, respiratory rate).
Reports	Free text reports of electrocardiogram and imaging studies.

3.2.2 Discharge summaries from MIMIC-III

The data used in this work is from previous work by Gehrmann, Sebastian, et al [17]. According to Sarmiento RF, DERNONCOURT F. [17], among all the data in the NOTES class, discharge summaries hold the most valuable information for patient phenotyping [47]. Therefore, we only focused on the discharge summaries in this thesis.

There are in total 52,746 discharge notes for 46,146 unique patients in MIMIC-III. Each note has a free-form discharge summary and unique identifiers which include subject ID, admission ID and chart time.

Table 3.2 shows 3 random examples from the patients' discharge notes. In the table, *HAdm.ID* is an unique hospitalization for a patient in the database. *Subject.ID* represents a unique patient in the database, hence by joining *HAdm.ID* and *Subject.ID* one can find the specific hospitalization for each patient. In the database one *Subject.ID* can be combined with different *HAdm.ID* because a patient can have different hospitalizations in different time. Cohort marks whether a patient is a frequent visitor (defined ≥ 3 ICU visits within 365 days) or not. Conditions fields contain the information about whether a patient has specific phenotypes (annotated by the experts, see next section) or not, and the last column is the discharge summary with respect to the corresponding hospital admission for a patient.

Table 3.2. 3 patient examples from discharge notes.

Items	HAdm.ID	Subject.ID	Chart.time	Cohort	Conditions	Discharge Summary
1	118003	3644	118003	1	1,0,0,0,0,0...	"Admission Date:"
2	137421	4074	137421	0	0,0,0,0,0,0...	"Admission Date:"
3	191406	3644	137421	1	1,0,1,0,0,0...	"Admission Date:"

The texts showed below is one discharge summary example from the database:

Admission Date: [**2151-7-16**]

Discharge Date: [**2151-8-4**]

Service:

ADDENDUM:

RADIOLOGIC STUDIES: Radiologic studies also included a chest CT, which confirmed cavitary lesions in the left lung apex consistent with infectious process/tuberculosis. This also moderate-sized left pleural effusion.

HEAD CT: Head CT showed no intracranial hemorrhage or mass effect, but old infarction consistent with past medical history.

ABDOMINAL CT: Abdominal CT showed lesions of T10 and sacrum most likely secondary to osteoporosis. These can be followed by repeat imaging as an outpatient.

[**First Name8 (NamePattern2) **]
 [**First Name4 (NamePattern1) 1775**]
 [**Last Name (NamePattern1) **], M.D. [**MD Number(1) 1776**]

Dictated By:[**Hospital 1807**]
 MEDQUIST36
 D: [**2151-8-5**] 12:11
 T: [**2151-8-5**] 12:21
 JOB\#: [**Job Number 1808**]

In all the summaries, some personal sensitive information such as names and dates are masked by the de-id process. The de-id process will not be discussed in this thesis since its not related to our work.

3.2.3 Annotated dataset

There are in total 1,610 annotated samples. The samples were taken from all the discharge summaries by firstly extracting 415 ICU frequent visitors, and 313 randomly selected summaries from the same frequent visitors in the 415 patients but their later visits. Also 882 patients summaries were randomly selected from those who were not frequent visitors. So there are in total 1,610 summary notes. All the 1,610 notes were annotated into 10 different phenotypes. Each patient can be annotated with multiple phenotypes, also each summary was annotated at least twice for each phenotype. The annotators include 7 people which consist of two clinical researchers, two junior medical residents, two senior medical residents and a practicing intensive care medicine physician. The positive samples in each phenotype ranges from 126 to 460 cases. Cohen's Kappa (κ) measure was used to show the inter-rater agreement in each phenotype. When there is disagreement in annotating between two different annotators, one of the senior clinicians will decide on the final label. Table 3.3 shows the numbers and percentages about the positive samples and the kappa coefficients [39] for each phenotype.

Table 3.4 shows the numbers of patients that have certain numbers of phenotypes.

Table 3.3. Numbers and percentages of positive samples for 10 different phenotypes.

Phenotype	positive samples	κ
Adv. Metastatic Cancer	161 (10.00%)	0.83
Adv. Heart Disease	275 (17.08%)	0.82
Adv. Lung Disease	167 (10.37%)	0.81
Chronic Neurologic Dystrophies	368 (22.85%)	0.71
Chronic Pain	321 (19.93%)	0.83
Alcohol Abuse	196 (12.17%)	0.86
Substance Abuse	155 (9.62%)	0.86
Obesity	126 (7.82%)	0.94
Psychiatric disorders	295 (18.32%)	0.91
Depression	460 (28.57%)	0.95

Table 3.4. Numbers of patients that have certain numbers of phenotypes.

Occurrences of Phenotypes	Number
Patients with 0 phenotypes	359
Patients with 1 phenotypes	545
Patients with 2 phenotypes	345
Patients with 3 phenotypes	207
Patients with 4 phenotypes	113
Patients with > 5 phenotypes	41

Cohen's Kappa (κ)

Cohen's Kappa is a measurement that evaluates the inter-rater agreement for categorical items. It takes the possibility of the agreement into consideration, hence resulting a more robust measurement for agreement evaluation [39].

Let's assume there are in total 50 samples and 2 raters. The raters need to decide if a sample is good or not. a are the total samples that both raters agree to be good, b are the total samples first rater measures to be good while second says to be bad, c are the samples first rater says bad while second says good, and d are the samples both agree to be bad. Then the observed proportionate agreement is calculate by the following equation 3.7 (assuming $a = 30$, $b = 20$, $c = 20$, $d = 30$).

$$p_0 = \frac{a + d}{a + b + c + d} = \frac{60}{100} = 0.6 \quad (3.7)$$

One needs to consider the random agreement chance which can be calculated by the follow equation 3.8.

$$p_{\text{good}} = \frac{a + b}{a + b + c + d} \cdot \frac{a + c}{a + b + c + d} = 0.5 * 0.5 = 0.25 \quad (3.8)$$

The probability of random disagreement is

$$p_{\text{bad}} = \frac{c + d}{a + b + c + d} \cdot \frac{b + d}{a + b + c + d} = 0.5 * 0.5 = 0.25 \quad (3.9)$$

The final Cohen's kappa can be calculated by

$$\kappa = \frac{p_0 - p_{\text{good}} + p_{\text{bad}}}{1 - p_{\text{good}} + p_{\text{bad}}} = \frac{0.6 - 0.25 + 0.25}{1 - 0.25 + 0.25} = 0.2 \quad (3.10)$$

3.3 Processing pipeline

The samples extracted from the MIMIC-III are stored in a comma-separated values (csv) file, and the discharge summary fields are string format. Discharge summaries are unstructured data, although most of the medical related terms are coded according to ICD-9, ICD-10, or Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT) [25]. The overall writing is full of spellings errors, typing errors, acronyms, and abbreviations. Meanwhile the writing styles may vary a lot according to personal preferences. Therefore, a natural language processing pipeline is needed before inputting the data into neural network. There are two preprocessing flows for our experiment, one is for word level preprocessing, while the other one is for sentence level preprocessing. In this subsection we will introduce word level preprocessing first.

3.3.1 Preprocessing for word level input

In our experiment, the preprocessing was done using python. The data set we used contained 1,610 total samples, and each sample contained normally a different length of string characters, the data can be expressed as $Data = \{S_1, S_2, S_3, S_4, \dots, S_{1610}\}, S_n \in [String]$.

Word level preprocessing processes the whole texts of a sample, thus the sentence or paragraph structures are not considered in this case. Sentence level preprocessing will be introduced later in this section.

An overall pipeline of word level preprocessing can be seen in the figure 3.10.

Cleansing

Cleansing steps aim to remove all the meaningless characters regarding our model. In this thesis, we followed the preprocessing step in previous work [17]. Characters for alphabets and numbers, and some special characters like , () ! ? ' were kept, other characters were removed, since other characters were considered useless in our model other than those characters that we kept.

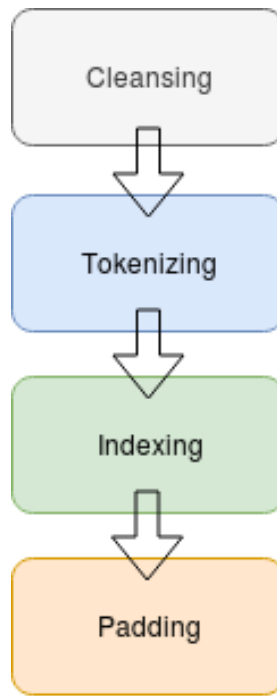


Figure 3.10. Overall pipeline of word level preprocessing of the data.

We used python script to replace all the characters to be removed with a single empty space (including empty space itself), and multiple continuous targets were replaced with a single empty space. This was important because when we were converting each token to its integer representation, the algorithm recognized each individual token by spaces.

An example of this process can be seen in the figure 3.11.

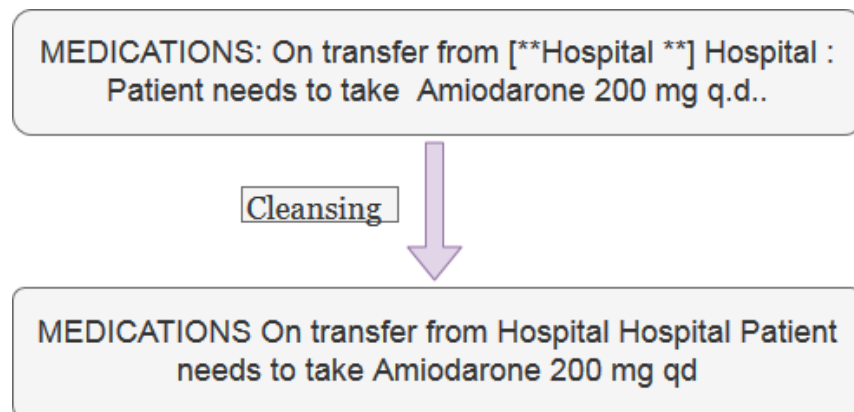


Figure 3.11. An example of in total 2 sentences being cleansed (texts are taken from part of a sample just for illustration).

From the figure 3.11 one can see that, special characters like ':', '[', '**', '.' were all removed, and the sequences of special characters like '[**' were also removed. After removing sequence '**]' there were two spaces between two 'Hospital' words in this example, and these two spaces were replaced with a single space.

3.3.2 Tokenization, indexing, padding

Tokenization is the process of setting the boundaries for each individual unit (token). An token can be a word, a sequence of numbers, or a special character. After tokenization, each token was recorded into a lookup dictionary with the index indicating its position in the dictionary. The order of the dictionary can be arbitrary, but in our experiment the dictionary was constructed according to the order of the appearances of the tokens in the samples. The index starts from 1, 1 is the index for token 'unk', which is assigned to the token that appeared in the samples but did not exist in the pretrained word embedding. 2 is the index for 'padding' token, which is used to pad the samples to a fixed length. There were in total 48,848 different tokens in our experiments, hence the size of the lookup dictionary was 48,848.

Tokenization was followed by indexing procedure which is to replace each token with its index in the lookup dictionary, then each token will be represented as a categorical value. These values are later used in the word embedding lookup step to be replaced by their corresponding word embedding representations.

The last step is padding process. Due to the fact that lengths of the text in different samples may vary, and for the sake of convenience, all the samples with less than the maximum length were padded with integer 2 which indicates the token 'padding' in the lookup dictionary. The sample with longest length had 5,572 tokens. Therefore, all the other samples should be padded to have 5,572 tokens. The padding step is different for word and sentence level inputs, this will be discussed in the coming subsections.

Passing the samples through tokenization, indexing, and padding will produce a vector $S = \{s_1, s_2, s_3, s_4 \dots s_{1610}\}$, $s_n = \{x_1, x_2, x_3, x_4 \dots x_{5572}\}$, $x_n \in [1, 48848]$, and this is the structure of the input samples to the word level network.

An overall illustration can be seen from the figure 3.12.

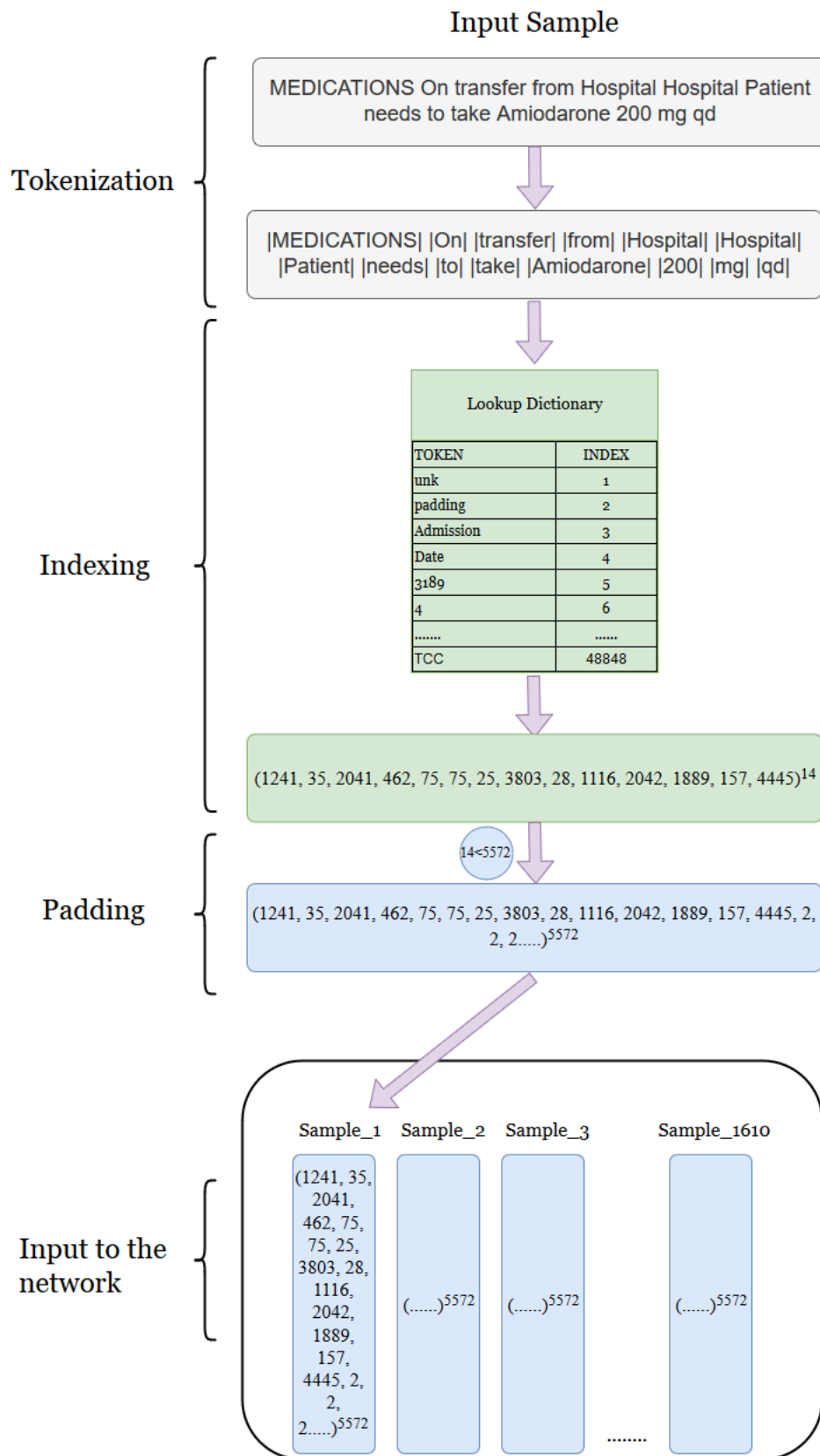


Figure 3.12. Illustration of tokenizing, indexing and padding of an input of in total 2 sentences (texts are taken from part of a sample just for illustration).

3.3.3 Preprocessing for sentence level input

In our work, we considered sentence level input in addition to word level input which consists of word embeddings, hence we used also input that comprises sentence embeddings (sentence embedding will be introduced later). The working pipeline for preprocessing sentences introduces one more step which is sentence segmentation. The pipeline is showed in the figure 3.13.

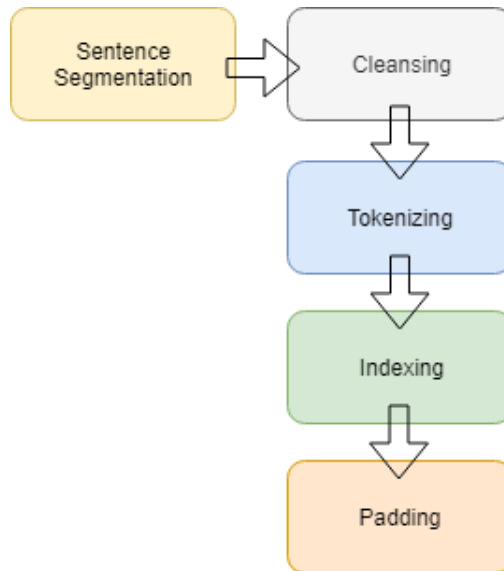


Figure 3.13. Overall pipeline of preprocessing sentence level input.

Sentence segmentation

Sentence segmentation is always a challenging step in preprocessing sentences, because in a free-form text, the sentence break does not always follow common rules. Some of the sentences are properly written and end with period, while rest of the sentences are not. In order to separate the sentences, we need to define some special separating rules to specify the boundaries of sentences besides using period as the general boundary. It is worthy to mention that, in the case of our work, we also considered comma as a boundary to be separated when the sentences between a comma both contained number of tokens greater than 5. The data to be processed has the same structure as the one in word level preprocess.

After specifying the boundaries, we trimmed the whole texts into different sentences for each sample, hence a sample was represented as a list of different sentences in our experiments. For comparing the structures of both level inputs, in the sample that contains word level inputs they are represented as

$Sample = \{Word_1, Word_2, Word_3, \dots, Word_n\}$, $Word_n \in [String]$. Instead, in the sample that contains sentence level inputs, the structure is

$Sample = \{Sentence_1, Sentence_2, Sentence_3, \dots, Sentence_n\}, Sentence_n \in [String]$

$Sentence = \{Word_1, Word_2, Word_3, \dots, Word_n\}, Word_n \in [String]$. Figure 3.14 shows a sentence segmentation example.

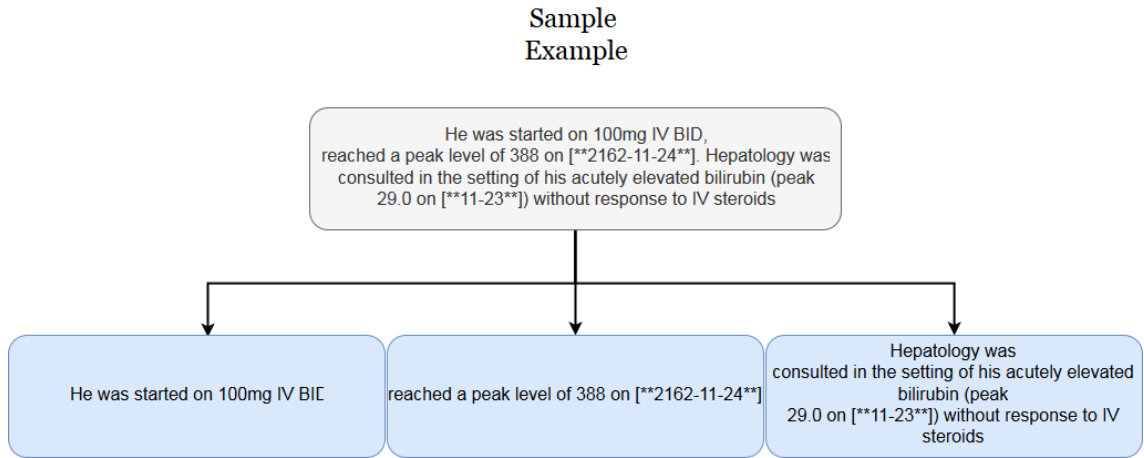


Figure 3.14. Illustration of sentence segmentation of a text sample. First sentence is separated by ',', second the third sentences are separated by '.' (texts are taken from part of a sample just for illustration).

The remaining preprocessing steps for sentence level input

Tokenization and indexing procedures are identical to the ones from word level preprocessing. For the padding step, each sentence may have different lengths of words, also each sample may have different lengths of sentences. Thus padding needs to be done in the sentence dimension as well as word dimension. In our work, the sentence with the maximum length among all the sentences has a length of 150, and the sample with most sentences has a number of 545 sentences. So in our experiment we padded all the sentences to have 150 tokens, and we used padding sentence which is an empty sentence (a sentence consists of 150 'padding' tokens) to pad into each sample until every sample has 545 sentences. An example can be seen in figure 3.15.

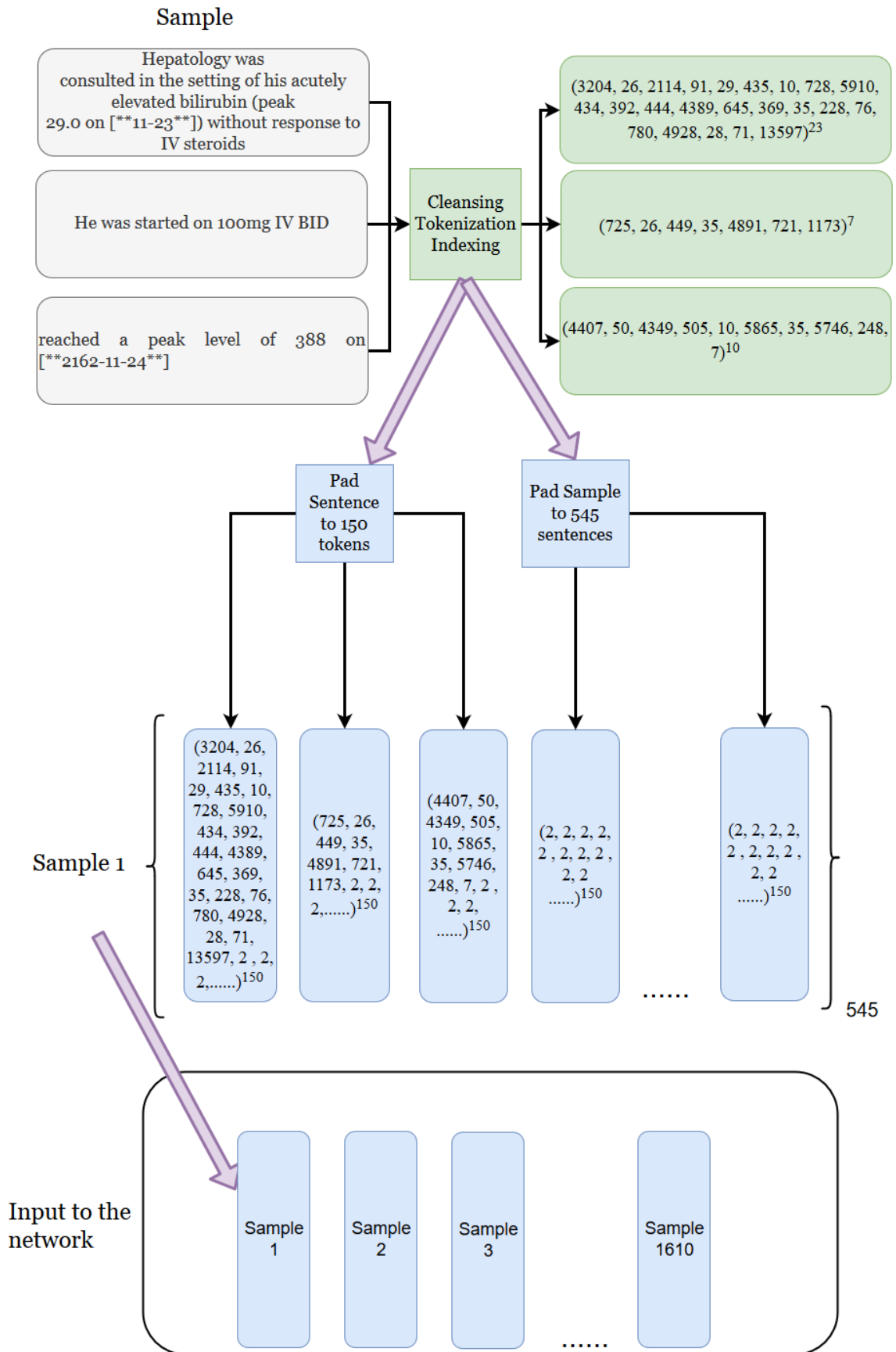


Figure 3.15. Illustration of a sample with 3 sentences being tokenized, indexed and padded (texts are taken from part of a sample just for illustration).

3.4 Word embedding

A number of natural language processing algorithms treat each word or token as a meaningless unit, and each unit is represented by a unique symbol. This type of technique is simple and robust [40]. One-hot vector and n-grams belong to this type of representation. Methods like these do not take the semantic and syntactic meaning of individual word into account. Therefore, they have limits in processing limited amount of data. Instead of these traditional ways there is also another representation method named word embedding. It refers to natural language processing approaches that combine language modeling and feature learning techniques to project the words to a distributed vector space where syntactic and semantic meanings of the words are captured. The idea of word embedding was proposed in 1986 by Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams [46]. Many approaches have been proposed regarding learning word embeddings [4, 40, 45], and models based on word representation learning methods were shown to outperform traditional models [4, 41].

In this thesis, we used the pretrained word embeddings learned using word2vec [40] approach. Word2vec is a well-known approach to train word embeddings, and it is relatively fast compared with other approaches while producing promising results. It has two basic models, continuous Bag-of-Words (CBOW) model and continuous Skip-gram model (SG). They both use neural network architecture with in total 3 layers. CBOW model tries to predict the current word based on its neighborhood words using a sliding window, while SG model predicts the surrounding words from the position of the input word also using a sliding window. The input to the network is a sequence of words, while the middle layer projects the corresponding words within the sliding window to the word embedding dimension, the output layer calculates the probability distribution for each word to appear in the corresponding position. The goal of the network is to optimize the parameters in the network. Those parameters are later used to construct the word embedding dictionary where different words can be mapped into their vector representations using their unique tokens to lookup from the dictionary.

Word embeddings are able to record the syntactic and semantic meaning of words in their embedding vectors. The captured meanings in the vectors even go beyond common syntactic regularities [40], which allows meaningful algebraic operations between words. For example, embedding vector for ("King") - embedding for ("Man") + embedding for ("Woman") approximates an embedding vector that is close to embedding for ("queen") in the word embedding space. Furthermore, operations like vector ("French") + vector ("actress") results in a vector close to vector ("Juliette Binoche") and other vectors of french actress. This characteristics enables the network to capture more abstract level of representations from input. Additionally, words that have similar syntactic or semantic meanings are mapped to an area in the embedding space close to each other. This property helps address the problem of spelling errors, spelling mistakes, abbreviations and synonyms which largely exist in the free-form texts.

An example of word embeddings representations is shown in figure 3.16.

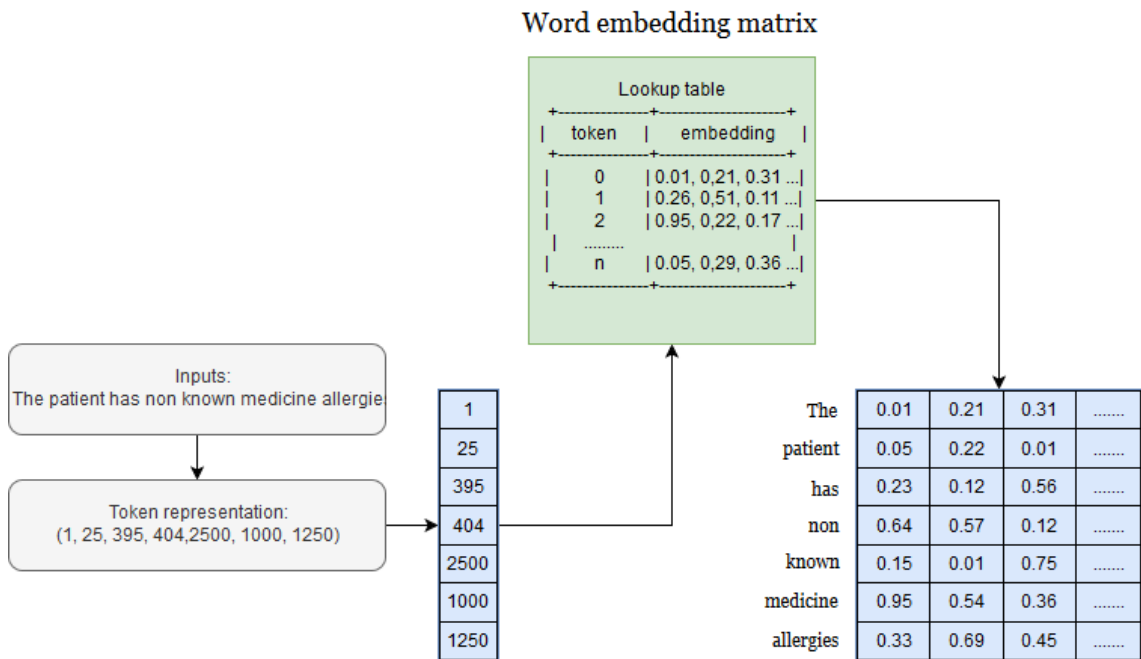


Figure 3.16. An example of how word embedding works.

The word embedding we used in this thesis was the same as the one from the previous work by Gehrmann, Sebastian, et al [17]. The word embeddings were trained on all the discharge notes from MIMIC-III database. The model used to train word embedding was CBOW model. The lookup window size was set to 10, and the model filtered out the words appearing lesser than 5 times. Also for every positive example 10 negative examples were sampled. The model was trained for 15 iterations. The total number of vocabularies in the word embedding lookup table that was trained on the whole discharges notes was 470,856, while the number of vocabularies for the 1,610 samples that we used in our experiment was 48,848. The word embeddings we used were case-sensitive. Therefore, the same words with upper or lower cases were assigned with different word embeddings.

3.5 Network architecture

In this work, we replicated the CNN architecture proposed by Collobert et al. [8] and Kim Yoon [30]. Additionally, we added another CNN that processes sentence level input.

3.5.1 Basic structure

The idea behind Kim Yoon's CNN architecture is that it tries to learn different features from the combinations of adjacent words using varying sizes of convolutional windows. Also the convolutional windows only operate along the temporal dimension because there

is no meaningful spatial dimension for text data. The basic structure of the CNN we used in this thesis is shown in figure 3.17.

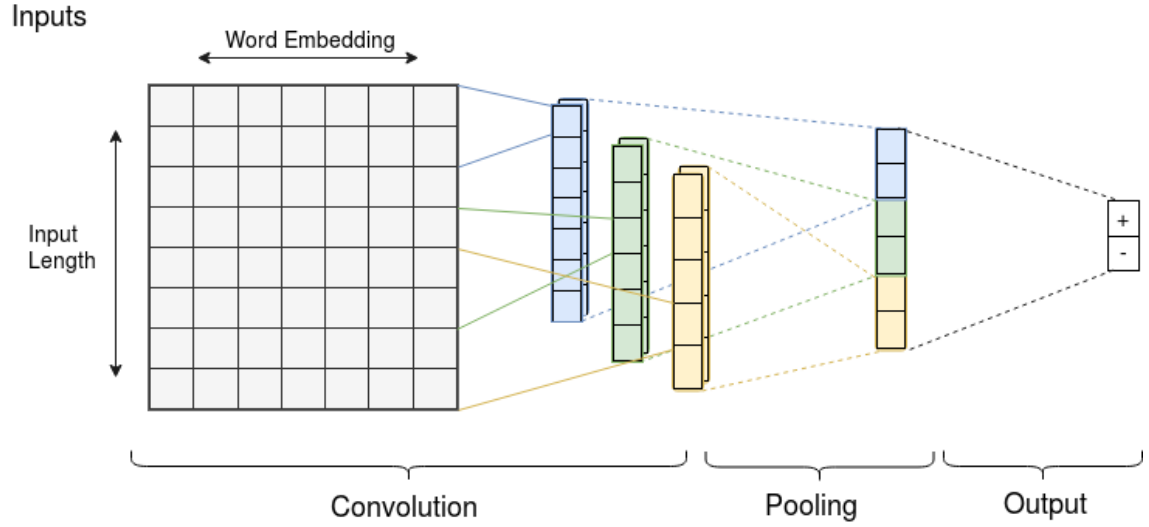


Figure 3.17. The basic structure of the CNN used in this thesis. Word level network and sentence level network are both based on this structure, but using different convolutional window sizes. In the figure, the lengths for the filters are 2,3,4, and the number of feature maps for the filters is 2.

In the network, the input is a matrix where vertical dimension is the length of either word or sentence, and horizontal dimension is the length of embeddings. Each row is represented as an embedding for a token or an embedding for a sentence. All the rows are represented as a sequence of embeddings. Each row inside the input is $x_i \in \mathbb{R}^{1 \times d}$, where d denotes the dimension of pretrained word embedding which was fixed to 50 in our experiments. Convolutional windows will go through the temporal dimension using a stride of 1. Convolutional windows have dimensions of $w_i \in \mathbb{R}^{i \times d}$, $i \in \{1, 2, 3, \dots, n\}$, here i denotes the filter length of the convolution window. Each convolutional operation between the filter window the receptive field will produce a single value inside the corresponding feature map. Different feature maps represent different relations extracted from the convolution operations over the neighborhood words. This is essentially a process of looking different n-grams from the input.

Various convolutional filters process at the same input data in parallel. Therefore, each group of convolution filters with the same length constitutes a single convolutional layer. Each convolution layer produce n feature maps, and each feature map has different lengths of values according to the filter size. Assume that the length of the filter window is l , then each feature value inside a feature map is calculated as $c_i = f(w \cdot x_{i:i+l-1} + b)$, here b denotes the bias term, $x_{i:i+l-1}$ denotes the corresponding region processed by the filter, f is the activation function. We used only ReLu in our experiment, the property of ReLu can be checked from table 2.1. Then each feature map will be $C_i = \{c_1, c_2, c_3 \dots c_{n-l+1}\}$, here i denotes the number of feature maps.

Followed by convolution layer is a max-over-time pooling layer. Max-over-time pooling

takes only the maximum value from a feature map, which results in a single value for each feature map. It is argued that getting the maximum activation values from the features maps is most important to the semantics in text mining [8]. During the pooling procedure all the values pooled from the feature maps are concatenated together to form a single vector. If there are number of L convolutional layers, each layer has n feature maps, after pooling, the penultimate layer will have a feature vector $V = \{v_1, v_2, v_3 \dots v_L \times n\}$, $v \in \mathbb{R}^1$, here v are the values pooled by max-over-time pooling.

The last layer consists of two neurons marking the final probability distribution of true and false labels. They are calculated using an activation function and the feature vectors from the previous layer. In this case output $y = \sigma(w \cdot V + b)$, here w is the parameters between the last and previous layer, b is the bias, V is the final feature vector. Since we were dealing with binary classification task, we choosed Sigmoid as the activation function σ , the property of Sigmoid can be seen from the table 2.1.

3.5.2 Details of the practical network

For our experiments, we used a framework that combines two CNNs processing at word and sentence level inputs independently. The penultimate layer from combined network concatenated features outputted from both networks, and the combined network will make predictions using these features. Figure 3.18 shows an overview of the combined network.

Word level network

For our word level network, we replicated the structure from Gehrmann, Sebastian, et al 2018 [17]. We used the same parameters and setup as theirs. The network receives preprocessed word level input where each sample is a vector of integers. In our case, the maximum length of text L is 5,572, each sample is $x = \{t_1, t_2, t_3 \dots t_{5572}\}$, $t \in [1, 48848]$. The first layer of the network is an embedding layer where each token is mapped into a vector of real numbers of dimension 50. The embedding layer is a two-dimensional matrix of dimensions 48848×50 and itself acts as the lookup table. Since the inputs are tokens of their indexes in the table, the embedding vectors for the inputs can be acquired using a simple indexing. In the embedding layer, the input x will be expanded into a 2D matrix $E \in \mathbb{R}^{5572 \times 50}$. All the weights in the embedding layer are trainable. The network will update these values each backward pass, consequently the embedding lookup table will also be updated. We found that keeping the embeddings trainable resulted in better performances than keeping them statistic in our experiments. After the embedding layer, the data is fed into the neural network, which is the same architecture as figure 3.17, but experimenting with different convolution filter sizes. An simple example of what word level input resembles can be seen in the word embedding figure 3.16.

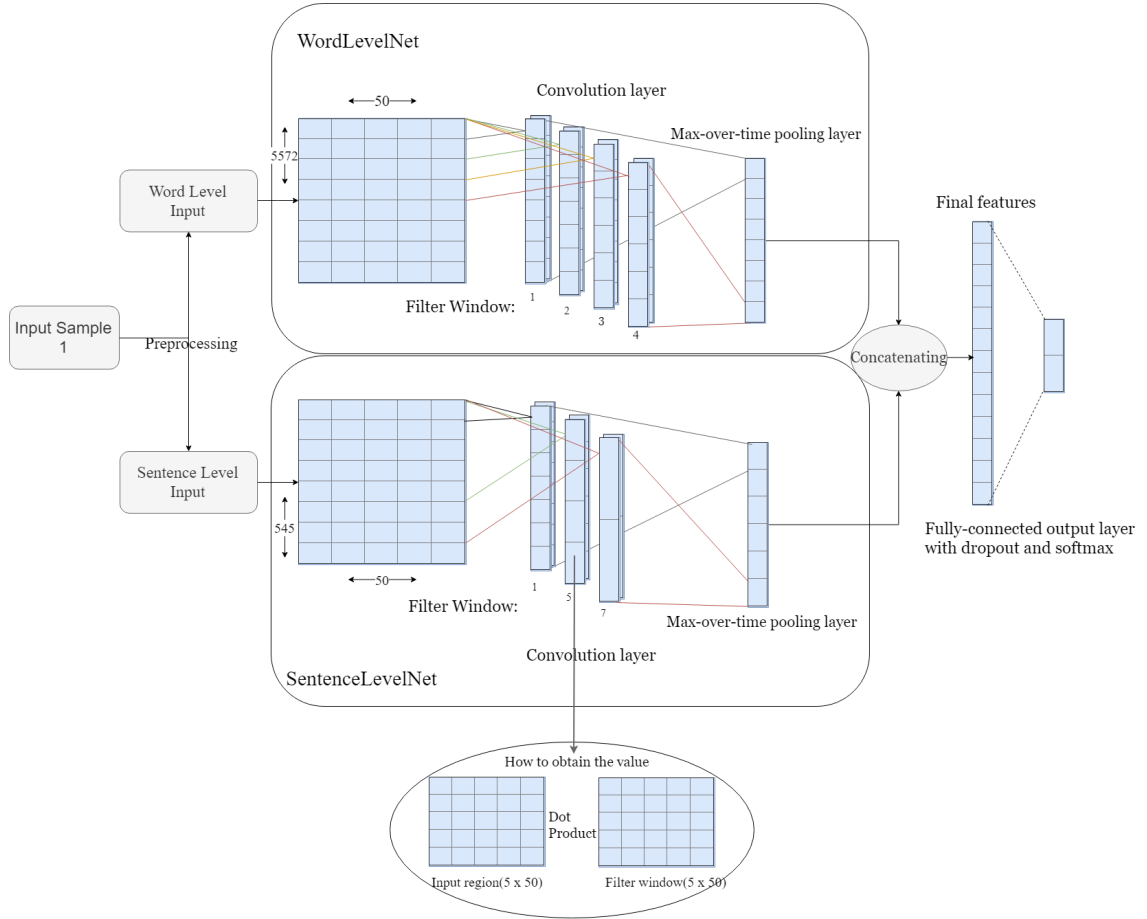


Figure 3.18. An overview of the combined network.

Sentence level network and sentence embedding

In the sentence level network, each sample has 545 sentence vectors, and each sentence vector has 150 tokens. The sentence level network receives the preprocessed sentence level input, where each sample $x = \{s_1, s_2, s_3 \dots s_{545}\}$, $s \in t^{1 \times 150}$, $t \in [1, 48848]$. Embedding layer is also applied to the sentence level input. In the embedding layer the input x will be transformed into a 3-dimensional matrix $E \in \mathbb{R}^{545 \times 150 \times 50}$. In order to utilize underlying features in different sentences in an efficient way, we pooled the matrix E by either averaging or summing along the second dimension. It is worthwhile to note that the padding tokens in each sentence will not contribute to the pooling operation. The pooling is calculated according to the original length for each sentence before padding. After pooling, the matrix for each sample will be $E \in \mathbb{R}^{545 \times 50}$, here 545 is the number of sentences. Each sentence in the sample was pooled to a single vector called sentence embedding that has dimension of 1×50 which is the same as a word embedding for a token. Thus we denote it as sentence embedding, and each sample in sentence level input is represented as a sequence of sentence embeddings. Figure 3.19 shows how we obtain sentence embeddings in a sample.

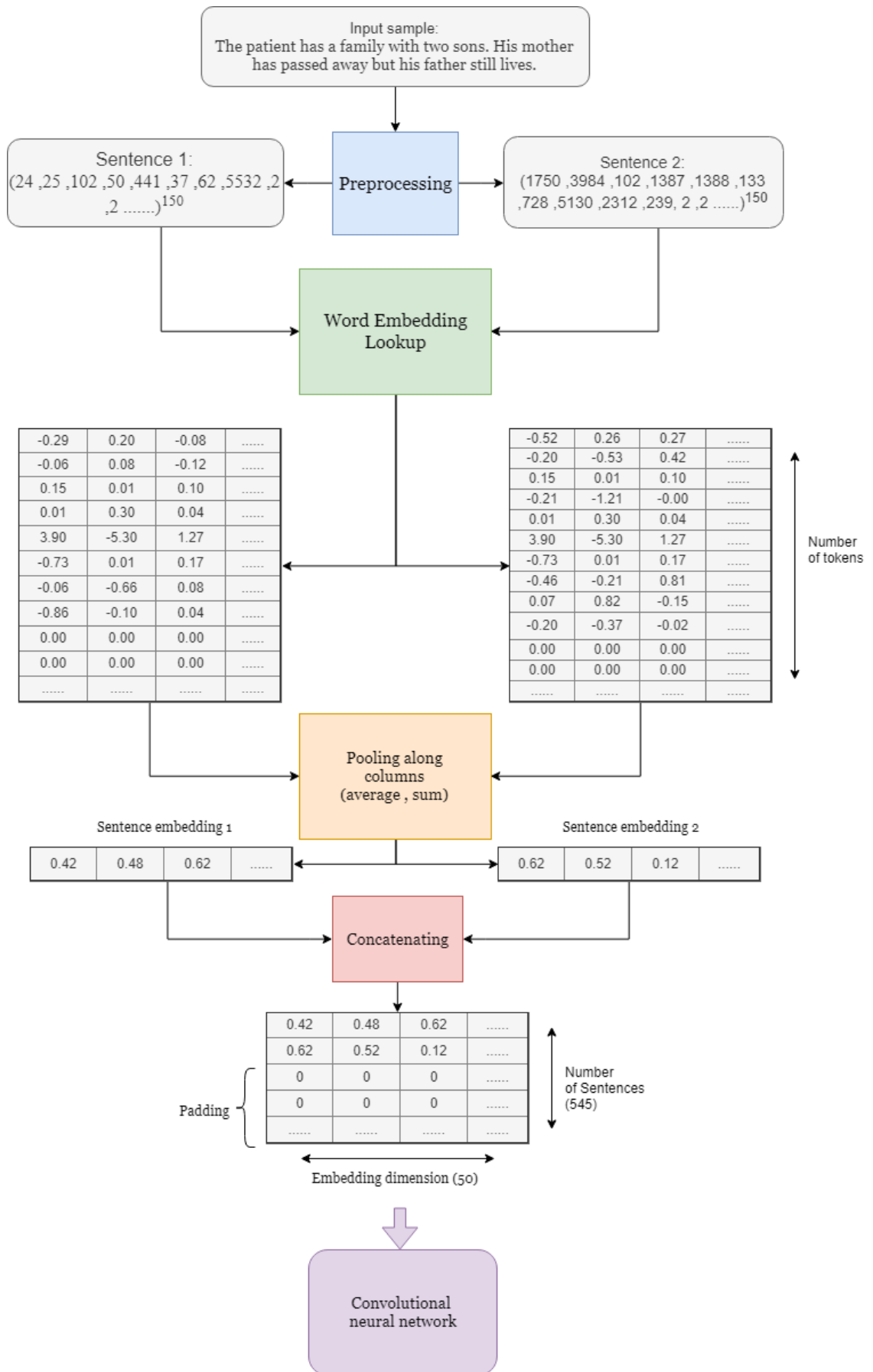


Figure 3.19. An illustration of how we produce sentence embeddings.

4 RESULTS AND DISCUSSION

In this chapter, we will show and discuss the results of our work. For all the experiments we have performed, the codes were built using python. The implementation, training and testing of the network used the pytorch package.

We firstly discuss the statistical analysis we performed on the data samples. Secondly, part we will report and discuss the network we replicated from previous work conducted by Gehrmann, Sebastian, et al [17], we will refer to this network architecture as word level network.

In the next section, we tested the network architecture that used both word and the sentence level input, and we refer to it as the combined network. This network architecture is the main work of this thesis. Then we compared the performance of our network architecture and the network architecture we replicated from previous work. We present the performance of each network architecture by their accuracy, specificity, precision, recall, ROCAUC and F1 score on 10 different phenotypes. All the results were reported using a 10-fold cross-validation with standard errors (marked with \pm). Precision, recall and F1 score were reported with respect to positive label.

For the last part we carried out several experiments on the data set to see how the network behaves with different inputs.

4.1 Statistical analysis on the data

In this section, we performed a statistical analysis on the data. The data we used contain 1,610 samples, and each sample corresponds to a specific discharge summary of a hospital admission for a patient. For the word level input, we found that the maximum length of a sample was 31,214 characters, which corresponds to 5,572 tokens. The average length of number of characters in each sample was 11,525, and the average length for number of words in each sample was 2,067. The sample with least length of characters had 361 characters. For the sentence level inputs, the average number of sentences in each sample was 210. The sample with least number of sentences had 8 sentences, and the one with most sentences had a number of 545. For all the sentences, the one had the longest length is 150 words, and the average length of a sentence was 9 words. Furthermore, we looked into the occurrences for each token from the total vocabulary which has 48,848 different tokens. The results are shown in the figure 4.1,

figure 4.2 and figure 4.3.

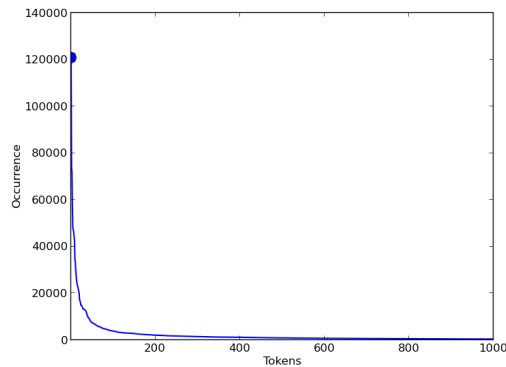


Figure 4.1. Number of Occurrences for the 1,000 most frequent tokens.

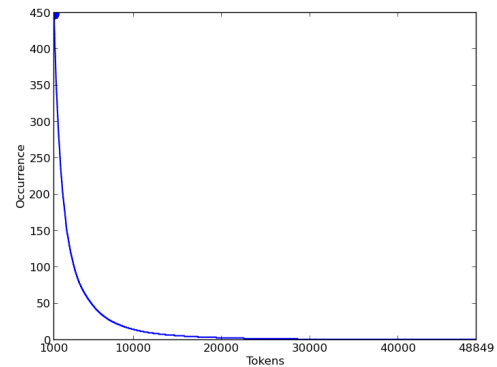


Figure 4.2. Number of Occurrences for the remaining tokens.

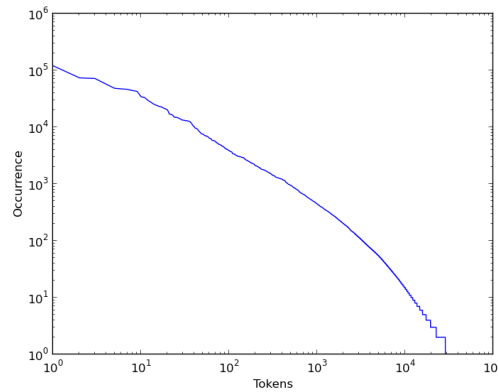


Figure 4.3. Logarithm scale of number of the occurrences of tokens from 1 to 48,848.

We showed the occurrences of tokens by three figures because of the huge difference in number of occurrences. One can see from the table there was roughly a number of 12 tokens that had frequencies more than 20,000, and most of the tokens had frequencies under 450. Table 4.1 shows the frequencies of tokens falling into different ranges.

Table 4.1. Numbers of tokens falling into different frequency ranges.

Frequencies	1	2-5	5-10	10-25	25-50	50-100	100-200	> 200
Numbers	20,293	12,807	4,103	4,065	2,362	1,947	1,280	1,990

One can tell that most of the tokens had frequencies from 2 to 50. But to our surprise that there were 20,293 tokens that only appeared once in the whole data set. As we investigated into these tokens, we discovered that most of them were unique sequences of numbers, such as '1203', and '11850'. The rest of them were some rare vocabularies, spelling errors, and different styles of abbreviations. Also there were tokens that were wrongly operated by the preprocessing steps, which resulted in words that start or end with symbols and words that do not exist in English vocabularies. Additionally, we found that even though the word embedding lookup table was obtained through training on all

the discharge notes inside MIMIC-III dataset and it has a number of 470,260 total different tokens, and during the preprocessing steps there were still 4,852 different tokens that did not exist in the pre-trained word embedding lookup table. We argue that this is because of the different preprocessing steps used in training word embeddings and cleansing the samples. These tokens that did not exist in the word embeddings were assigned with word embeddings initialed with random weights between -0.25 to 0.25.

Table 4.2 shows the 10 most frequent tokens from the samples. Table 4.3 shows the 10 most frequent words (tokens that are not conj, pronoun, preposition, article, numbers) from samples.

Table 4.2. The 10 most frequent tokens.

Token	Frequencies
,	120,749
)	73,927
(71,970
and	57,430
to	48,438
the	47,192
of	46,442
was	44,176
1	42,881
with	34,476

Table 4.3. The 10 most frequent words.

Token	Frequencies
mg	22,780
Tablet	21,073
patient	16,859
Name	13,357
padding	12,882
day	11,958
One	10,324
BLOOD	9,188
Last	8,361
Discharge	7,781

4.2 Network for word level input

In this section, we will report the result we obtained by training the network that used only word level inputs. The network was a replication from work [30], and was tuned according to previous work [17]. We reported the results on 6 different measurements: accuracy, specificity, precision, recall, AUC under ROC, and F1. The network architecture can be seen from figure 3.17.

In our work, each phenotype was trained individually. Therefore, we had 10 total different models for 10 phenotypes. The setting of parameters for all the models was the same. The weights for convolutional layers were initialised using a uniform distribution of range -0.01 to 0.01. We used the optimal convolutional filter lengths which were reported by the previous work [17] for different phenotypes. The filter lengths can be seen in table 4.4. Each filter group (refers to a group of filters with the same length) resulted in 100 feature maps. The parameters in the feature maps were normalized to a norm of 3, and the dropout rate was 0.5. We calculated the cross-entropy to estimate the loss, and we used Adadelta [58] as our optimizer, with a learning rate of 0.5. For more details of the training, we used a batch-size of 32 and ran each model with 20 epochs. The word

embedding weights were further fine-tuned with the gradients passed only from the word level network during backward pass.

Table 4.4. Different convolution filter lengths for different phenotypes in our experiments.

Filter lengths	1,2,3	1,2,3,4	2,3,4,5	1,2,3,4,5
Phenotypes	Substance Abuse	Adv. Cancer, Chronic Neuro, Depression	Adv. Lung Disease, Chronic Pain, Alcohol Abuse, Obesity, Psychiatric Disorders	Adv. Heart Disease

As for splitting the dataset, from the table 3.3, we can see that the samples were very imbalanced with labels. When doing 10-fold cross validation this could result in very different performances. In order to counter this problem, for each phenotype, we divided the dataset into 10 different subsets with equal sample size. Each subset contained equally distributed positive samples with respect to the corresponding phenotype. Therefore, we had 10 different data sets regarding 10 phenotypes. When we trained different models we used the corresponding data set that was equally sampled. Regarding training, we used 1 of the 10 subsets as the testing set, and rest of them together as the training set for each training process. The order of the 10 subsets were fixed. We performed the training process 10 times each time with a different test subset, and the results were averaged. Table 4.5 shows the results for the word level network. The results were reported using a 10-fold cross-validation with their standard errors which were marked by \pm .

From the table 4.5 we can see that the accuracy and specificity were very high for all the phenotypes, but the precision and recall were nowhere near the scores of these. This is due to the imbalanced between positive and negative samples. As we can see from the table 3.3 the positive samples ranged from 7.82% to 28.57% and had a average of 15.67%. As a result, we only focused on the scores of precision and recall, ROCAUC and F1 in this work. We noticed that for most of the cases we had a better precision over recall. This is also due to the imbalanced labels, but one can adjust the filter lengths to get results with different precision and recall while keeping the F1 score stay mostly the same. For our experiment we only showed the models trained with the hyper-parameters that gave the best F1 scores. We also observed that the standard errors we obtained from 10-fold cross validation were quite high for some situations. As you can see from the standard error for recall of advanced lung disease was ± 4.94 , and the standard errors were more obvious especially for the precision and recall than other metrics.

It is worthy to mention that we wanted to reduce the side effect of imbalanced labels as much as possible during training. So we passed the class weights to our loss function to make our learning cost-sensitive. We fed a vector to the cross-entropy loss function with the predefined weight scales regarding each phenotype. The value of the weight for each

Table 4.5. Results from word level network.

Phenotypes	Accuracy %	Specificity %	Precision %	Recall %	ROCAUC %	F1Score %
Adv. Cancer	95.27 ±0.58	98.05 ±0.34	80.11 ±3.49	70.62 ±3.05	84.34 ±1.54	74.45 ±2.40
Adv. Heart Disease	90.12 ±0.61	94.07 ±0.87	72.49 ±2.77	70.92 ±2.55	82.50 ±1.09	71.07 ±1.47
Adv. Lung Disease	91.86 ±0.57	95.77 ±0.43	61.18 ±2.93	58.04 ±4.94	76.89 ±2.43	58.90 ±3.68
Chronic Neuro	86.14 ±0.91	91.94 ±0.94	71.72 ±2.34	66.67 ±3.82	79.31 ±1.78	68.41 ±2.30
Chronic Pain	84.72 ±0.78	93.63 ±1.18	68.29 ±3.76	48.91 ±2.59	71.27 ±1.06	55.96 ±1.69
Alcohol Abuse	94.90 ±0.56	97.73 ±0.53	83.23 ±3.43	74.50 ±3.12	86.11 ±1.53	78.04 ±2.40
Substance Abuse	93.10 ±0.64	95.05 ±0.64	62.80 ±3.13	74.83 ±3.79	84.94 ±1.87	67.66 ±2.72
Obesity	97.57 ±0.40	98.99 ±0.28	87.80 ±3.59	81.02 ±3.86	90.00 ±1.95	83.72 ±2.85
Psychiatric Disorders	92.91 ±0.71	95.06 ±0.56	79.26 ±2.25	83.33 ±2.02	89.19 ±1.18	81.59 ±1.94
Depression	91.55 ±0.76	97.13 ±0.67	92.03 ±1.53	77.60 ±3.33	87.36 ±1.48	83.69 ±1.79

class was $1/classsamples$. This allows the network to penalize more if the network makes wrong prediction on the minority class. By applying this technique we saw significant improvements over the F1 score on three phenotypes, while not effecting much on other phenotypes. The results for these three phenotypes are reported in table 4.6.

Table 4.6. Results of models for 3 phenotypes before and after class weight scaling.

Phenotypes	Precision%		Recall%		ROCAUC%		F1%	
	Before	After	Before	After	Before	After	Before	After
Adv. Lung Disease	80.55 ±4.97	61.18 ±2.93	35.29 ±3.54	58.04 ±4.94	67.07 ±1.71	76.89 ±2.43	47.74 ±3.62	58.90 ±3.68
Chronic Neuro	75.66 ±2.61	71.72 ±2.34	55.80 ±4.63	66.67 ±3.82	75.00 ±2.02	79.31 ±1.78	62.79 ±3.09	68.41 ±2.30
Chronic Pain	77.36 ±5.34	68.29 ±3.76	35.82 ±4.27	48.91 ±2.59	65.93 ±1.53	71.27 ±1.06	45.92 ±2.75	55.96 ±1.69

4.3 Network for word and sentence level inputs

For this section, we are going to introduce and discuss the combined network that utilizes both word and sentence level inputs. In the combined network, it accepted an extra input comprised of individual sentence embedding. The preprocess and the procedure regarding how we obtained sentence embedding is described in the section 3.3.3 and section 3.5.2. The combined network stacked two neural networks that each process word and sentence level separately. It combined the features outputted from both networks together to make predictions. The structure is illustrated from figure 3.18. The word level network in the combined network used the same input, parameters and filter window lengths from the section 4.2 to keep the result convenient for comparing. The sentence network also used the same hyper-parameters except for the filter window lengths and the number feature maps. For the sentence network we used in our experiment, the filter window lengths for all the phenotypes were fixed to 1,5,7,9, and the number of feature maps was set to 50 for each filter group. The embeddings were shared between both word and sentence level network, but only the network that accepts word level input will pass gradients to update the word embeddings during training. This helped to hugely reduce the training time of the combined network. The results of the combined network are reported in the table 4.7.

Table 4.7. Results from combined network.

Phenotypes	Accuracy %	Specificity %	Precision %	Recall %	ROCAUC %	F1Score %
Adv. Cancer	95.03 ± 0.63	97.23 ± 0.48	75.53 ± 3.62	76.69 ± 3.08	86.96 ± 1.56	75.37 ± 2.10
Adv. Heart Disease	91.36 ± 0.73	95.17 ± 1.04	77.67 ± 3.23	72.83 ± 2.86	84.00 ± 1.17	74.36 ± 1.44
Adv. Lung Disease	92.11 ± 0.57	96.18 ± 0.03	63.41 ± 3.46	56.83 ± 4.51	76.51 ± 2.25	59.34 ± 3.57
Chronic Neuro	86.21 ± 0.82	91.78 ± 1.06	71.72 ± 2.22	67.46 ± 3.25	79.62 ± 1.45	68.91 ± 1.88
Chronic Pain	85.03 ± 1.11	91.69 ± 1.76	67.11 ± 4.21	58.22 ± 1.97	74.96 ± 0.53	61.78 ± 1.25
Alcohol Abuse	95.59 ± 0.54	98.37 ± 0.44	87.40 ± 3.40	75.52 ± 3.15	86.95 ± 1.59	80.59 ± 2.49
Substance Abuse	93.41 ± 0.77	95.46 ± 0.65	64.71 ± 4.05	74.16 ± 3.61	84.81 ± 1.92	68.68 ± 3.39
Obesity	97.63 ± 0.41	99.05 ± 0.30	88.63 ± 3.78	81.02 ± 3.69	90.04 ± 1.87	84.13 ± 2.85
Psychiatric Disorders	94.03 ± 0.70	96.35 ± 0.58	83.96 ± 2.37	83.66 ± 1.79	90.00 ± 1.08	83.72 ± 1.90
Depression	91.98 ± 0.51	96.95 ± 0.90	92.04 ± 1.97	79.56 ± 2.38	88.26 ± 0.90	84.94 ± 1.09

We used summing and averaging pooling for forming the sentences embeddings. Con-

sidering that different pooling methods can result in different performances. We only picked the pooling method that gave the best results on F1 score. The pooling methods we used for different phenotypes can be seen in the table 4.8.

Table 4.8. *Different pooling methods for different phenotypes.*

Pooling methods	Summing	Averaging
Phenotypes	Adv. Cancer, Adv. Heart Disease, Chronic Pain	Adv. Lung Disease, Chronic Neuro, Alcohol Abuse, Substance Abuse, Obesity, Psychiatric Disorders, Depress

4.4 Comparing the two networks

The performances of word level network and combined network will be discussed in this section. Firstly we reported the precision, the recall, F1 and ROCAUC scores attained by both network architectures in one table 4.9 for comparison.

We can observe that there was an overall improvement over all the phenotypes in F1 score which we think its the most critical score for this task. The percentage gain in F1 scores after adding a sentence network ranged from 0.41% to 5.82% and had an average of 1.8%. For ROCAUC, we also observed improvements on some of the phenotypes, while rest of them remained unchanged. Additionally, adding sentence network resulted in slightly differences in the balance between precision and recall scores for some phenotypes, i.e., advanced cancer, advanced lung disease and chronic pain, but globally the F1 scores improved and we did not see any drawbacks in the performances for this task.

According to Mikolov, Tomas, et al [42], word embedding representations allow meaningful algebraic operations between word embeddings. As they reported, the vector of token "French" + the vector of token "actress" approximates vectors of "Juliette Binoche", "Cecile De" or "Charlotte Gainsbourg", and this is what motivated us to perform this experiment. As the previous work from Gehrmann, Sebastian, et al [17] showed, sometimes adding a bigger filter window did not help the performance of certain phenotypes, and even hurt the performances. However, each sample can contain thousands of tokens, and the inability to search for bigger neighborhood of words will limit the ability of the network to understand the documents. Therefore, we introduced the sentence embedding, which is done by pooling all the tokens in a sentence. This will result in a single vector that we call sentence embedding. The procedure of producing a sentence embedding is essentially mapping the whole sentence to the embedding space by algebraic operations between all the words in a sentence. The results we obtained somewhat showed that these extra information can be utilized by the network to get better results, because the network can better understand the whole documents using information from both words and sentences instead of just considering neighborhoods of maximum 5 words.

Table 4.9. Results from both network architectures (*W* denotes word level network, *W+S* denotes combined network).

Phenotypes	Precision%		Recall%		ROCAUC%		F1%	
	W	W+S	W	W+S	W	W+S	W	W+S
Adv. Cancer	80.11 ±3.49	75.53 ±3.62	70.62 ±3.05	76.69 ±3.08	83.34 ±1.54	86.96 ±1.56	74.45 ±2.40	75.37 ±2.10
Adv. Heart Disease	72.49 ±2.77	77.67 ±3.23	70.92 ±2.55	72.83 ±2.86	82.50 ±1.09	84.00 ±1.17	71.07 ±1.47	74.36 ±1.44
Adv. Lung Disease	61.18 ±2.93	63.41 ±3.46	58.04 ±4.94	56.83 ±4.51	76.89 ±2.43	76.51 ±2.25	58.90 ±3.68	59.34 ±3.57
Chronic Neuro	71.72 ±2.34	71.72 ±2.22	66.67 ±3.82	67.46 ±3.25	79.31 ±1.78	79.62 ±1.45	68.41 ±2.30	68.91 ±1.88
Chronic Pain	68.29 ±3.76	67.11 ±4.21	48.91 ±2.59	58.22 ±1.97	71.27 ±1.06	74.96 ±0.53	55.96 ±1.69	61.78 ±1.25
Alcohol Abuse	83.23 ±3.43	87.40 ±3.40	74.50 ±3.21	75.52 ±3.15	86.11 ±1.53	86.95 ±1.59	78.04 ±2.40	80.59 ±2.49
Substance Abuse	62.80 ±3.13	64.71 ±4.05	74.83 ±3.79	74.16 ±3.61	84.94 ±1.87	84.81 ±1.92	67.66 ±2.72	68.68 ±3.39
Obesity	87.80 ±3.59	88.63 ±3.78	81.20 ±3.86	81.02 ±3.69	90.00 ±1.95	90.04 ±1.87	83.72 ±2.85	84.13 ±2.85
Psychiatric Disorders	79.26 ±2.25	83.96 ±2.37	83.33 ±2.02	83.66 ±1.79	89.19 ±1.18	90.00 ±1.08	81.59 ±1.94	83.72 ±1.90
Depression	92.03 ±1.53	92.04 ±1.97	77.60 ±3.33	79.56 ±2.38	87.36 ±1.48	88.26 ±0.90	83.69 ±1.79	84.94 ±1.09

Efficiency comparison

In our experiment, we stacked another network to the original architecture, which means that we essentially added more parameters and an extra input to the network. This will no doubt influence the efficiency of the network. Table 4.10 shows the parameters in two network architectures and the running time for an epoch for them. The testing environment is Windows 10, with CPU Intel i5-6300HQ 2.30Ghz and GPU Nvidia GTX 970M.

Table 4.10. Efficiency comparison of two network architectures. The ranges for the parameters correspond to different convolution filter lengths in the network.

Network types	Trainable parameters(embedding)	Trainable parameters(network)	Running time(epoch)
W network	2,442,400	30,600 to 76,000	10.44s
W+S network	2,442,400	86,000 to 131,400	11.79s

4.5 Experiments on removing less frequent tokens from dataset

According to the table 4.1, we saw that the number of tokens that only appeared once in the data set is 20,293, which was almost 40% of the total tokens. Most of these tokens were spelling errors, different kinds of abbreviations, and wrongly preprocessed words which can be words that start or end with symbols. Also there was a large number of tokens that appeared less than 10 times in the total data set. Hence, we argue that the tokens will contribute to the network performance only after a certain threshold of frequencies. We performed experiments to see how the model performed when we remove the tokens that are less or equal than frequency 0, 1, 5, 10, 25, 50, 100, 200. The numbers of total tokens to be removed with respect to the removed tokens' frequencies are listed in table 4.11.

Table 4.11. *The numbers of total tokens removed with respect to those removed tokens' frequencies.*

Occurrences	<=1	<=5	<=10	<=25	<=50	<=100	<=200
Numbers	20,293	33,102	37,205	41,270	43,632	45,579	46,859

We performed this experiment on three phenotypes. We only reported the F1 score and the standard errors and we used only the word level network with the same parameters used in word level network testing. The results were reported using a 10-fold cross-validation with standard errors which were marked \pm .

The results for psychiatric disorders can be seen from figure 4.4. From the figure we saw that when we removed tokens with occurrences lesser or equal than 1, the network slightly improved. When we further removed more tokens with occurrences lesser equal than 25 we saw a slight drop in the performance, but until this point the F1 score almost stayed the same. However, when we started to remove tokens with occurrences greater than 50 the network began to drop rapidly in performance until it hit 65% F1 score when we removed up to 46,857 tokens. The best performance we observed is the network trained by the data set where tokens with occurrences less equal than 1 were removed, which is reasonable according to our experiments since these tokens were mostly the noises from the samples. When we tried to further remove more frequent tokens, we saw only harm to the performance of the network.

We performed the same test on obesity, and the result can be seen in the figure 4.5. As we can see, we had some similar results as psychiatric orders. The performance stayed stable around 83% until we started to remove tokens with frequencies ≥ 50 , and we started to see significant drawbacks in the performance.

However, when we employed this test on depression phenotype. We observed some counter-intuitive results shown in figure 4.6. According to the figure the best result we attained was the model trained with data set where 37,203 tokens were removed. How-

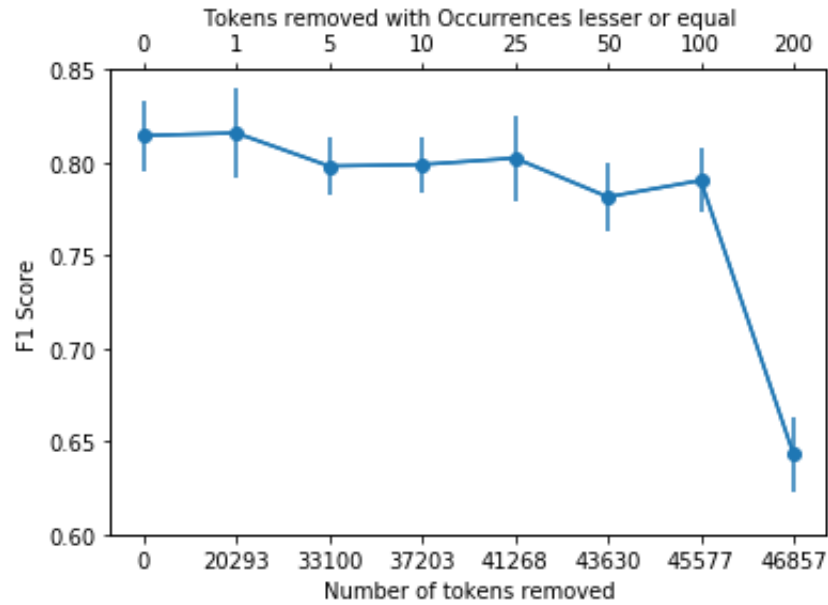


Figure 4.4. F1 scores of models of psychiatric disorders trained with data set where a certain number of tokens were removed.

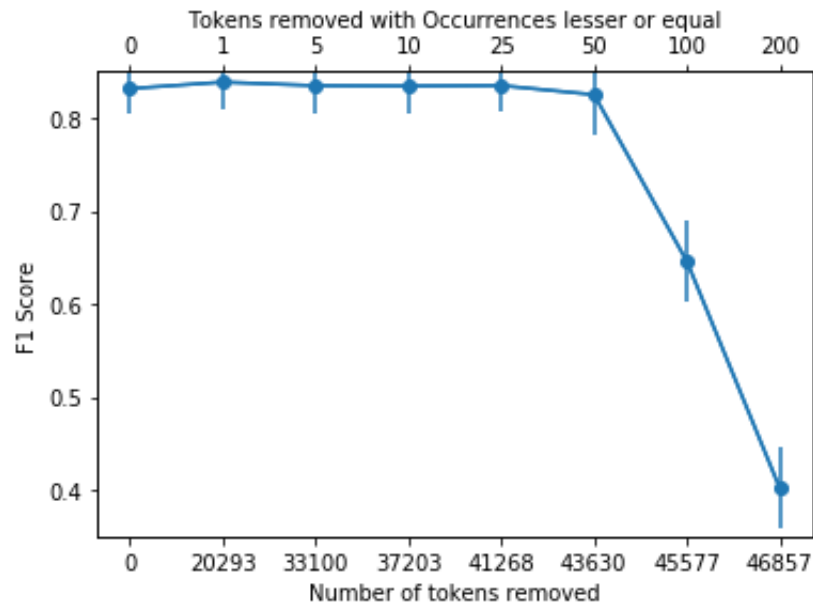


Figure 4.5. F1 scores of models of obesity phenotype trained with data set where a certain number of tokens were removed.

ever, to our surprise, as we kept on trimming tokens up to in total 46,857, the models behaved stably regardless how many tokens we removed, and all of the models trained with trimmed data sets outperformed the original model that unitized all the vocabularies.

To conclude, we would like to suggest that it is helpful to remove tokens with frequencies ≤ 1 , since most of these tokens are noises. Tokens that only appeared once in the data set are extremely difficult to make predictions based on them as they either appear in

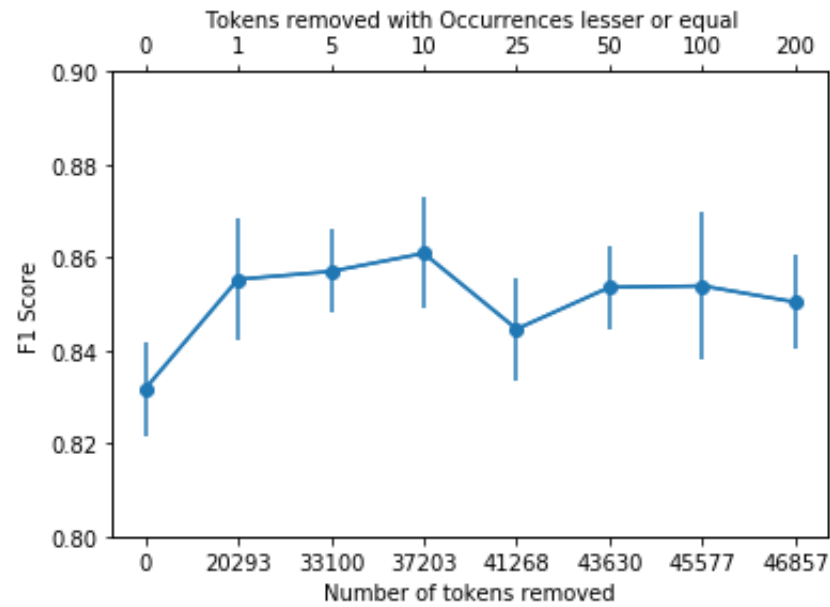


Figure 4.6. *F1 scores of models of depression phenotype trained with data set where a certain number of tokens were removed.*

training, or testing set, hence it is impossible to evaluate them. Furthermore, we argue that it is rather safe to remove tokens ≤ 10 or even ≤ 25 frequencies according to the results we observed. But the situation depends on different phenotypes. Never the less, we will not recommend to remove tokens ≥ 25 frequencies because that is where the data set starts to lose key information for prediction and models will have difficulties understanding the data.

5 CONCLUSION

In recent years we have seen a dramatic increase in the adoption of electronic health records. An increasing number of hospitals have their own electronic health systems. We can correctly derive patient phenotypes by analyzing rich information stored in the electronic health records. This can be essential for performing patient care related tasks in hospitals. EHRs contain varying important information for patient phenotypes, especially the texts recorded in the digital clinical narratives. These unstructured texts are typically harder to be processed than the structured information in eHRs, but the free-form text can hold much valuable information regarding patient phenotypes that structured data can not capture, therefore, making the unstructured data usable through machine learning and natural language processing is what we have done in this thesis.

Researches have applied traditional machine learning algorithms in cooperation with natural language processing techniques in mining clinical notes from eHRs. Those methods have yielded good results, but in recent years deep learning algorithms have been carried out in many aspects of natural language processing tasks including mining medical records yielding better results than traditional machine learning methods. Deep learning algorithms have shown their superiority over traditional methods by their ability to learn the abstract representations from the raw text with minimized human intervention instead of using heavy predefined medical terms by the experts.

In this thesis, we applied CNN which is one of the algorithms in deep learning. CNN is expected to be good at extracting local features from the texts, which means that it can detect meaningful information from the combinations of neighborhood words, and utilize this information in predicting certain phenotypes. In the previous work [17], they used 10 CNN models trained on 10 different phenotypes. The network architecture utilized convolutional windows up to lengths of 5, and they showed that this architecture was able to outperform some of the traditional machine learning methods. By looking into the previous work, we discovered that the downside of this architecture is the inability of searching for larger local regions of the documents. The network was by structure limited to combinations of words up to 5 lengths. Consequently in this thesis, We modified the architecture by feeding an additional sentence level input to an additional CNN that can process this level input. This network can produce features by considering the relations across many sentences, thus taking relations existing in larger regions of the texts into accounts when making decisions.

Our results have shown that the architecture that combines two networks that process

word and sentence level inputs separately was able to produce better results. From our experiments, we have observed by F1 score that our network architecture constantly outperformed the network architecture that only uses word level inputs. The biggest margin of improvement was by 5.82% in chronic pain phenotypes, which was the hardest one for CNN to make decisions on. This was followed by 3.29% in advanced heart disease, 2.55% in alcohol abuse, and 2.13% in psychiatric disorders. In other phenotypes, we only saw small margins of improvement in the F1 scores. Furthermore, there were also improvements in the ROCAUC scores for some phenotypes. We saw improvement in ROCAUC scores from 83.34% to 86.96% in advanced cancer, from 71.26% to 74.96% in chronic pain, and from 82.50% to 84.00% in advanced heart disease. In contrast, about 1/3 of the phenotypes did not benefit from the extra sentence level network. We argue this is a result of the poor preprocessing step and the pooling methods we used.

What motivated us to consider using sentence embedding is the paper from Mikolov, Tomas, et al. They have shown that word embeddings allow meaningful algebraic operations between vectors. When we pooled the all the words in a sentence, we were equivalently performing algebraic operations between all the words and essentially mapping the sentences into the embedding dimension. In order to obtain sentence level input, we defined some general rules to set the boundaries for separating sentences in the whole documents. This was a sophisticated process because in free-form texts the writing formats and styles can be very different according to different recorders, therefore, the quality of each separated sentence was not guaranteed and we could not go through all the sentences in the training set for checking. By examining into the sentences in some samples, we argue that, about 30% of the separated sentences were clear and meaningful, 40% of them could make senses on their own, while 30% of them still remained to be better processed. Additionally, we acquired the sentence level input by pooling all the word embeddings in a sentence. Typically in a sentence, not all of the words contribute to the sentence, words like 'a', 'the', and some unique sequences of numbers. They are considered as meaningless tokens when we pool the sentence. This will effect the quality of the sentence embeddings we get after pooling. The way we obtained sentence embeddings is by pooling the all the words in a sentence. We consider pooling solution an efficient way to preserve the overall meanings of the sentences, however, by summing up or taking averaging for all the word embeddings in the sentence, we sometimes lose much information regarding each sentence, especially when a sentence is long. The factors mentioned above could be the reasons why adding an extra sentence level network did not effect on some phenotypes.

Overall, adding an extra sentence level network only increased 10% total training time in the original network, while providing more potential for the model to better utilize the local information between words as well as the local information between sentences. We consider this solution beneficial to experiment on, but one should consider better preprocessing steps for separating the sentences in a document and set up better rules to eliminate the possible noises for forming sentence level inputs in order for one to see significantly better outcomes.

In addition to the evaluation of the networks, we also performed some analyzes on the data set, and it helped us to get some insights from the data we used. We observed that, in the original data set, there were 20,293 tokens that only occurred once, and 37,203 tokens only appeared lesser than 10 times in the total samples. Therefore, we argue that the token will contribute to the performance of the network only after a certain occurrences. We evaluated the models that trained on data sets with certain numbers of tokens removed for 3 phenotypes. We found out that all of them benefited when the tokens with 1 occurrence were removed, also they had better performances when tokens with 10 occurrences were removed, however removing tokens with > 25 occurrences might hurt the performances of the models in most of the cases.

To conclude, the network architecture we proposed in this thesis suggests that it has the potential to better understand the documents by taking both local information between words and local information between sentences into account. Also the extra network only increased 10% of the total training time. Meanwhile, we also found that the the distribution of occurrences of tokens in the data set we used in this thesis was unbalanced, thus, the data set contained a number of tokens that could be removed while not effecting or even improving the performances of the deep learning models that we used.

For our implementation in this work, there are still many future works left to be done to further improve the efficiency and accuracy. For example, using other pooling methods that can preserve as much information as possible, better preprocessing on the sentence level input, or trying other network variants such as recurrent neural network for the sentence level input.

REFERENCES

- [1] C. Angermueller, T. Pärnamaa, L. Parts and O. Stegle. Deep learning for computational biology. *Molecular systems biology* 12.7 (2016), 878.
- [2] A. R. Aronson. Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. *Proceedings of the AMIA Symposium*. American Medical Informatics Association. 2001, 17.
- [3] M. F. Azam, A. Musa, M. Dehmer, O. P. Yli-Harja and F. Emmert-Streib. Global Genetics Research in Prostate Cancer: A Text Mining and Computational Network Theory Approach. *Frontiers in genetics* 10 (2019), 70.
- [4] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research* 3.Feb (2003), 1137–1155.
- [5] G. S. Birkhead, M. Klompas and N. R. Shah. Uses of electronic health records for public health surveillance to advance public health. *Annual review of public health* 36 (2015), 345–359.
- [6] D. Charles, M. Gabriel, T. Searcy et al. Adoption of electronic health record systems among US non-federal acute care hospitals: 2008-2012. *ONC data brief* 9 (2013), 1–9.
- [7] Z. Che, Y. Cheng, Z. Sun and Y. Liu. Exploiting convolutional neural network for risk prediction with medical feature embedding. *arXiv preprint arXiv:1701.07474* (2017).
- [8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research* 12.Aug (2011), 2493–2537.
- [9] M. Denil, A. Demiraj, N. Kalchbrenner, P. Blunsom and N. de Freitas. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830* (2014).
- [10] J. C. Denny, N. N. Choma, J. F. Peterson, R. A. Miller, L. Bastarache, M. Li and N. B. Peterson. Natural language processing improves identification of colorectal cancer testing in the electronic medical record. *Medical Decision Making* 32.1 (2012), 188–197.
- [11] F. Doshi-Velez, Y. Ge and I. Kohane. Comorbidity clusters in autism spectrum disorders: an electronic health record time-series analysis. *Pediatrics* 133.1 (2014), e54–e63.
- [12] F. Emmert-Streib and M. Dehmer. A Machine Learning Perspective on Personalized Medicine: An Automized, Comprehensive Knowledge Base with Ontology for Pattern Recognition. *Machine Learning and Knowledge Extraction* 1.1 (2018), 149–156.

- [13] F. Emmert-Streib and M. Dehmer. Defining data science by a data-driven quantification of the community. *Machine Learning and Knowledge Extraction* 1.1 (2019), 235–251.
- [14] F. Emmert-Streib, S. Moutari and M. Dehmer. A comprehensive survey of error measures for evaluating binary decision making in data science. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2019), e1303.
- [15] F. Emmert-Streib, S. Moutari and M. Dehmer. The process of analyzing data is the emergent feature of data science. *Frontiers in Genetics* 7 (2016), 12.
- [16] C. Friedman. Medlee-a medical language extraction and encoding system. *Columbia University, and Queens College of CUNY* (1995).
- [17] S. Gehrmann, F. Dernoncourt, Y. Li, E. T. Carlson, J. T. Wu, J. Welt, J. Foote Jr, E. T. Moseley, D. W. Grant, P. D. Tyler et al. Comparing deep learning and concept extraction based methods for patient phenotyping from clinical narratives. *PloS one* 13.2 (2018), e0192360.
- [18] J. Geraci, P. Wilansky, V. de Luca, A. Roy, J. L. Kennedy and J. Strauss. Applying deep neural networks to unstructured text notes in electronic medical records for phenotyping youth depression. *Evidence-based mental health* 20.3 (2017), 83–87.
- [19] M. van Gerven and S. Bohte. *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.
- [20] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071* (2014).
- [21] A. Graves, A.-r. Mohamed and G. Hinton. Speech recognition with deep recurrent neural networks. *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, 6645–6649.
- [22] K. Häyrynen, K. Saranto and P. Nykänen. Definition, structure, content, use and impacts of electronic health records: a review of the research literature. *International journal of medical informatics* 77.5 (2008), 291–304.
- [23] K. He, X. Zhang, S. Ren and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 770–778.
- [24] M. Jaderberg, K. Simonyan, A. Zisserman et al. Spatial transformer networks. *Advances in neural information processing systems*. 2015, 2017–2025.
- [25] P. B. Jensen, L. J. Jensen and S. Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics* 13.6 (2012), 395.
- [26] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi and R. G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific data* 3 (2016), 160035.
- [27] R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058* (2014).
- [28] N. Kalchbrenner, E. Grefenstette and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

- [29] D. Kalra and D. Ingram. Electronic health records. *Information technology solutions for healthcare*. Springer, 2006, 135–181.
- [30] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
- [31] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [32] L. A. Knake, M. Ahuja, E. L. McDonald, K. K. Ryckman, N. Weathers, T. Burstain, J. M. Dagle, J. C. Murray and P. Nadkarni. Quality of EHR data extractions for studies of preterm birth in a tertiary care center: guidelines for obtaining reliable data. *BMC pediatrics* 16.1 (2016), 59.
- [33] A. Krizhevsky, I. Sutskever and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012, 1097–1105.
- [34] I. J. Kullo, K. Ding, H. Jouni, C. Y. Smith and C. G. Chute. A genome-wide association study of red blood cell traits using the electronic medical record. *PloS one* 5.9 (2010), e13011.
- [35] Y. LeCun, Y. Bengio and G. Hinton. Deep learning. *nature* 521.7553 (2015), 436.
- [36] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86.11 (1998), 2278–2324.
- [37] M. Lin, Q. Chen and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400* (2013).
- [38] M. Liu, E. R. McPeck Hinz, M. E. Matheny, J. C. Denny, J. S. Schildcrout, R. A. Miller and H. Xu. Comparative analysis of pharmacovigilance methods in the detection of adverse drug reactions using electronic medical records. *Journal of the American Medical Informatics Association* 20.3 (2012), 420–426.
- [39] M. L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica* 22.3 (2012), 276–282.
- [40] T. Mikolov, K. Chen, G. Corrado and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
- [41] T. Mikolov, A. Deoras, S. Kombrink, L. Burget and J. Černock. Empirical evaluation and combination of advanced language modeling techniques. *Twelfth Annual Conference of the International Speech Communication Association*. 2011.
- [42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*. 2013, 3111–3119.
- [43] R. Miotto and C. Weng. Case-based reasoning using electronic health records efficiently identifies eligible patients for clinical trials. *Journal of the American Medical Informatics Association* 22.e1 (2015), e141–e150.
- [44] T. B. Murdoch and A. S. Detsky. The inevitable application of big data to health care. *Jama* 309.13 (2013), 1351–1352.

- [45] J. Pennington, R. Socher and C. Manning. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, 1532–1543.
- [46] D. E. Rumelhart, G. E. Hinton, R. J. Williams et al. Learning representations by back-propagating errors. *Cognitive modeling* 5.3 (1988), 1.
- [47] R. F. Sarmiento and F. Dernoncourt. Improving Patient Cohort Identification Using Natural Language Processing. *Secondary Analysis of Electronic Health Records*. Springer, 2016, 405–417.
- [48] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler and C. G. Chute. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association* 17.5 (2010), 507–513.
- [49] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [50] Q. Suo, F. Ma, Y. Yuan, M. Huai, W. Zhong, A. Zhang and J. Gao. Personalized disease prediction using a cnn-based similarity learning method. *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2017, 811–816.
- [51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, 1–9.
- [52] D. Tang, B. Qin, X. Feng and T. Liu. Effective LSTMs for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100* (2015).
- [53] N. P. Tatonetti, P. Y. Patrick, R. Daneshjou and R. B. Altman. Data-driven prediction of drug effects and interactions. *Science translational medicine* 4.125 (2012), 125ra31–125ra31.
- [54] N. T. Vu, H. Adel, P. Gupta and H. Schütze. Combining recurrent and convolutional neural networks for relation classification. *arXiv preprint arXiv:1605.07333* (2016).
- [55] E. W. Weisstein. Convolution. *MathWorld* 1.1 ().
- [56] H. Xu, S. P. Stenner, S. Doan, K. B. Johnson, L. R. Waitman and J. C. Denny. MedEx: a medication information extraction system for clinical narratives. *Journal of the American Medical Informatics Association* 17.1 (2010), 19–24.
- [57] W. Yin, K. Kann, M. Yu and H. Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923* (2017).
- [58] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv:1212.5701 arXiv preprint* (2012).
- [59] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557* (2013).
- [60] L. Zhou, J. M. Plasek, L. M. Mahoney, N. Karipineni, F. Chang, X. Yan, F. Chang, D. Dimaggio, D. S. Goldman and R. A. Rocha. Using Medical Text Extraction, Reasoning and Mapping System (MTERMS) to process medication information in out-

patient clinical notes. *AMIA Annual Symposium Proceedings*. Vol. 2011. American Medical Informatics Association. 2011, 1639.

- [61] L. Zhou, A. W. Baughman, V. J. Lei, K. H. Lai, A. S. Navathe, F. Chang, M. Sordo, M. Topaz, F. Zhong, M. Murrall et al. Identifying Patients with Depression Using Free-text Clinical Documents. *Studies in health technology and informatics* 216 (2015), 629–633.